Reversible Parallel Discrete-Event Execution of Large-scale Epidemic Outbreak Models

Kalyan S. Perumalla, Sudip K. Seal
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
perumallaks@ornl.gov, sealsk@ornl.gov

Abstract

The spatial scale, runtime speed, and behavioral detail of epidemic outbreak simulations altogether require the use of large-scale parallel processing. Here, an optimistic parallel discrete event execution of a reaction-diffusion simulation model is presented. Rollback support is achieved with the development of a novel reversible model that combines reverse computation with a small amount of incremental state saving. Parallel speedup and other runtime performance metrics of the system are tested on a small (8,192-core) Blue Gene / P system, while scalability is demonstrated on 65,536 cores of a large Cray XT5 system. Scenarios representing large population sizes (up to several hundreds of millions in the largest case) are exercised.

1. Introduction

The enormity of epidemic outbreak effects and the world-wide attention to controlling them are common knowledge, often appearing in daily news. While many non-technical factors appear in effectively dealing with them, an important technical aspect continues to be elusive and remains to be explored, namely, gaining a good understanding of epidemic dynamics and the ways and means by which various contributing factors affect its dynamics. Public health planners and policy makers use epidemiological simulations to study a variety of factors that influence epidemic dynamics within a population. For example, recently, Gojovic *et al* [1] reported the use of such simulations to demonstrate the effect of timely delivery of vaccinations on the attack rate of H1N1.

Aside from analytical models based on simplifications, simulation continues to be an important tool. In contrast to numerical integration-based analysis of analytical (differential equation-based) epidemic models, simulation often provides flexibility in incorporating many factors. Large spatial scales and high behavioral detail contribute to the challenge of sustaining simulations of epidemic propagation dynamics at the scale of cities, states, and countries. Certain epidemics with global spans may even serve to motivate simulations at world-scale.

Ideally, epidemiological models should be detailed enough to capture realistic scenarios and produce actionable insights. Realistic models tend to be very complex and the associated parameter space very large. In turn, this implies that the resulting computational problem becomes very large and unsuitable for sequential execution.

Decisions and policies are typically based on statistical inferences from results of multiple simulation runs that attempt to explore the model's associated parameter space as exhaustively as possible. This requires very fast turnaround times for each run so that enough statistics can be gathered within a reasonable duration of wall clock time to base actionable decisions on. In light of the above, the need for parallel execution of such epidemiological models becomes evident and, not surprisingly, large scale computational epidemiology has become an area of active research in recent years, particularly, in an era when larger and more powerful parallel platforms are becoming increasingly common. Developing scalable algorithms for large scale realistic epidemiological simulations that can exploit the computing resources offered by today's state-of-the-art parallel platforms is, therefore, imperative and an important need of the hour.

Here, we present a simulation development and scaling effort, based a novel reversible parallel discrete event formulation, of a large class of epidemics that can be modeled using reaction-diffusion dynamics. We focus on the computational problem of sustaining simulations of large-scale epidemic propagation scenarios, and examine the performance effects of implementation on very large computational platforms containing 10^3-10^5 processor cores. We present some of the first results in demonstrating parallel discrete event simulations (PDES) of such epidemics scaling to a large number of processors.

1.1. Related Work

The epidemic modeling literature is vast, starting with classical differential equations hat appeared in the 1920s for aggregate phenomena such as critical thresholds and herd immunity, to articles appearing as recently as 2009 on phenomena such as emergence of critical thresholds on a variety of small-world and scale-free social networks [2-6]. In the past few years, agent-based simulations were used by the National Institutes of Health's Models of Infectious Disease Agent Study (MIDAS) group to shape avian flu policy[7, 8]. Other methodologies such as patch models, distance-transmission models and multi-group models are discussed in [9]. A new class of reaction-diffusion based disaggregate models [10, 11] has emerged in the past few years, to facilitate planning by federal agencies (DHS, DoD and NIH).

Although historically, epidemic simulations are sequentially executed, parallel simulation has been used in the late nineties and mid 2000's to accommodate increasingly larger sizes of epidemic networks, and also speed up the execution to meet real-time constraints. Within parallel simulations, time-stepped approaches have been used to execute certain complex models at large-scale. The SPaSM simulator [12] is an example of such a scalable framework that has been successfully used for analyzing very large scenarios, including certain country-scale pandemic flu propagation analyses. In scenarios involving highly varying time-scales and non-uniform inter-person interaction structures, however, discrete event simulation can offer faster advances in simulation time. On the other hand, parallel discrete event simulation offers its own challenges: the need to reformulate the dynamics in terms of discrete events, and the need to reduce runtime overheads to make it possible to scale execution to large numbers of processors. The most closely related work is the very interesting use of a customized style of discrete event processing akin to optimistic simulation (albeit, on shared memory machines) for simulating millions of light-weight Java-base agents [13]. Their runtime uses a specialized algorithm highly optimized to exploit a simplified state machine of individual's infection states, and hence does not generalize to more complex scenarios and models. A city-scale small-pox spread simulation was reported in [14], with the focus more on interoperability of simulator modules, less on scalability and efficiency. Other, peripherally related, work includes Monte Carlo simulation models (e.g., [15]).

Systems such as EpiSimdemics, EpiSims, and EpiFast from Virginia Tech [16, 17] have advanced the state of the art in epidemics analyses greatly in the past few years. We have been directly motivated by their article in Supercomputing 2008 that reported scalability of execution on up to 512 processor cores, encountering certain synchronization problems beyond that scale. An enhanced version of their work to include models of exogenous interventions [17] also scales to about 250 processors.

Our current work borrows their reaction-diffusion model, but explores an alternative execution method distinct from their parallelization approach which could be roughly classified as functional parallelism. We apply the latest PDES methodologies to explore the potential of PDES executions to well beyond a thousand processor cores.

1.2. Our Approach

Here, we develop a parallel discrete event formulation of the reaction-diffusion model reported in [16]. We retain the power and flexibility afforded by their individual models, which are nearly agent-based in modeling power (with parameters tunable down to each individual). We apply optimistic parallel execution techniques with a view to accommodating the low-lookahead conditions that may be present in certain scenarios. For rollback, we employ a combination of state saving and reverse computation. For scalability, we utilize supercomputing platforms to increase the speed of simulation of large-scale scenarios. Our goals of this effort are not only to provide PDES alternatives to the currently best known results which are non-PDES-based, but also to achieve new levels of scale (hundreds of millions of individuals) simulated on many thousands of processor cores.

1.3. Contributions

This article makes multiple contributions. First, our PDES-based formulation of reaction-diffusion models, (presented in greater detail later in the article), is unique in that host mobility, intra-host state evolution, and inter-host reactions are all modeled in full, discrete event fashion. The time-scales are completely decoupled from each other, and do not require any a priori determination of a time step and/or a minimum time increment, unlike prior formulation in [16]. Our model also enhances the mobility model of [16] by introducing arbitrary travel time delays across locations. In terms of the interaction between the model and the simulation, our development and use of a reversible execution for epidemics is unique. In terms of scalability, the results reported here are among the first scalable PDES execution of epidemic models. In fact, to the best of our knowledge, the largest run of the simulation scenario on 65,536 processor cores of Cray XT5 reported in Section 4 may be the largest

processor-count to date of any non-trivial PDES application reported in the literature. It surpasses the PDES-based radio signal propagation model execution reported on 5,000 processor cores in [18].

1.4. Organization

The rest of the document is organized as follows. The forward model is described in Section 2. The reverse model is described in Section 3. A performance study is presented in Section 4. The article is summarized in Section 5, followed by Section 6 outlining some of our ongoing work.

2. Forward Execution Model

The model and the details of the reaction-diffusion system are described in this section, along with their discrete event execution formulation.

2.1. Model based on Reaction-Diffusion

A large class of epidemics may be modeled using a combination of reaction and diffusion processes. The reactions arise as a result of inter-entity interactions, (e.g., physical proximity for influenza). The dynamic chances for interactions arise as a result of mobility of the entities (e.g., meetings and other co-located activities). Geographical diffusion of entities facilitates chances for interactions among varying sample sets of entities.

Figure 1 shows a schematic for the reaction-diffusion abstraction on a system of individuals. Interaction points called "locations" demarcate the extents of interaction that individuals within that location incur during the time they are present at that location. Multiple locations are contained within a region. An arbitrary diffusion network may be defined among locations within a region. Similarly, an arbitrary diffusion network may be defined among regions. The diffusion network can be dynamic in nature; in other words, the connectivity graph may be changed at the modeler's/user's will at runtime, as needed by any specific scenario being modeled. Latencies of activities (interaction and/or mobility) along the links of the network may be arbitrarily specified.

Reaction-diffusion based models typically consist of: (a) a set of interacting entities, such as humans, (b) a network graph that represents the interaction structure of the population under study, (c) a reaction process, called within-host progression, that models the evolution of the disease within an individual entity in the population, and (d) a diffusion process, called between-host transmission, that models the transmission of the disease between individual entities. Other sub-models may be added to accommodate additional aspects such as resource limitations on all processes.

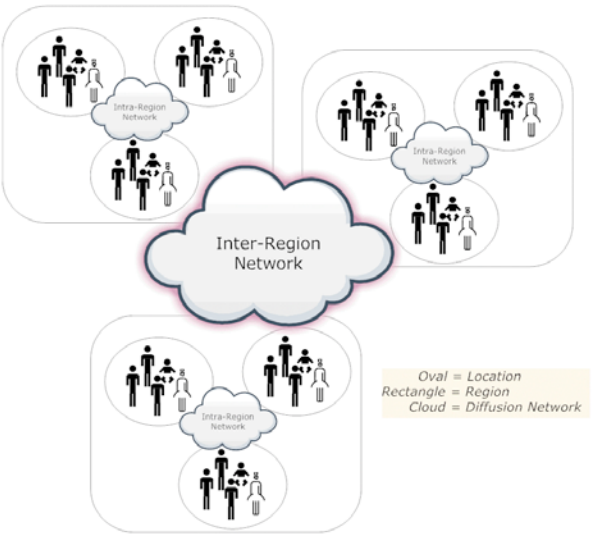

Figure 1: Abstraction of System for Epidemic Propagation

A reaction function defines the infection behavior of co-located individuals (probability of an uninfected entity getting infected, or, alternatively, an infectious individual infecting other susceptible population).

The within-host progression of the infection is modeled by a probabilistic timed transition system (PTTS) which is a finite state machine with probabilistic, timed state transitions. A PTTS reflects the reaction that changes the state of an individual within the population. We use the PTTS framework of [16] unmodified.

The between-host transmission is modeled by the following probability function:

$$p_i = 1 - e^{\tau \sum_{r \in R} N_r ln(1 - r s_i \rho)} \qquad \textit{(Equation 1)}$$

In the preceding equation, i is the label of an individual, τ is the duration of exposure, R is the set of infectivities of the infected individuals, N_r is the number of infectious individuals with infectivity r, s_i is the susceptibility of i and ρ is the transmissibility which is a disease specific property.

The underlying social network consists of the following entities defined hierarchically as: (a) individual person (b) location: a set of individuals (c) region: a set of locations and (d) domain: a set of regions. The scale of the system depends on the definitions that we associate with the above entities. For example, a state level simulation can associate locations with office buildings or shopping malls, regions can be associated with subdivisions, towns or

cities and the state itself as a group of regions which represents the computational domain over a state-level social network. Similarly, a country-level simulation can be appropriately redefined.

In summary, the system consists of (a) individual entities each with its own characteristics, (b) locations that hold individuals for periods of time, (c) regions that represent a set of logically related locations, who have a diffusion network among themselves, (d) a reaction function that specifies the probabilities of infections over time within a location, and (e) a two-level diffusion function that takes an individual from a location-region pair to another location-region pair.

In a simplified network, the following parameters may be defined: (i) *infected.prob*: percentage of population who are infected at time t=0 (ii) *vaccinated.prob*: percentage of population who are vaccinated at time t=0 (iii) *mean.staytime*: average time that an individual stays within a location (iv) *mean.localtraveltime*: average time taken by an individual to travel between two locations within a region (v) *mean.remotetraveltime*: average time taken by an individual to travel between locations across two regions and (vi) *locality.prob*: probability with which an individual stays within a region.

2.2. Relation to Real World Entities

The abstraction can be used to model activity at multiple spatial scales. If the number of persons per location is S_l and the number of locations per region is L_r, then, the following table illustrates the rough scales of population for which epidemic outbreaks could be analyzed using this model.

Table 1: Example Scales of Systems

S_l	L_r	*Scenario*
10	100	Small community
100	100	Small town
1000	100	Small city
1000	1000	Large city
10000	100	Rural state

The model is sufficiently flexible to accommodate intra-city and inter-city trips (work, stores, homes, schools), and varying levels of exposure times for reactions. In the largest case, one can envision modeling inter-country transport (ship, air), in combination with intra-country (inter-state) diffusion (car, air). Trip statistics obtained from the Bureau of Transportation Statistics are typically utilized to initialize the network structure and time distributions.

2.3. Modeling Operations and Policies

The same abstract model is in fact sufficiently powerful to model complex operations and policies. For example, curfews can be modeled by using an outgoing probability of unity and incoming probability of zero for the location corresponding to the curfew (e.g., a public park). Similarly, an outgoing probability of zero and incoming probability of unity may be used in the diffusion network to model quarantined locations (e.g., a school or hospital).

Analogously, it is possible to incorporate vaccination production delays (manufacturing latencies, vehicular transportation times) by setting or varying diffusion network latencies accordingly.

2.4. Our Discrete Event Model

As mentioned previously, we use a purely discrete event model. Each location is mapped to a single logical process (LP); a number of locations (specified by the user at runtime) constitute a region, and each region is mapped to a processor. The location LP processes three types of events, namely: (a) Arrival Event (*AE*) (b) State Change Event (*SCE*) and (c) Departure Event (*DE*). Each individual person in a location is assigned a globally unique integer identity to distinguish it within the entire population. A customized PTTS (specifiable by the user on a individual-basis at initialization) is adopted *a priori* that models the within-host progression of the infection. The state composition of each person and each location (LP) are given in Figure 2 and Figure 3 respectively.

Person State
ptype: Person Type and/or Traits
pid: Global Identifier
seed: RNG Seed
istate: Infection State

Figure 2: State Encapsulated by every Person

Location Data Structures (LP State)
occ: Occupant Set of Persons (Person States)
omap: Map from ID to Person in *occ* Set
gptts: Default PTTS across local occupants

Figure 3: Data Structures within each Location (LP)

The forward execution algorithm is shown in Figure 4. When an individual arrives, an *AE* is triggered in that location. As part of processing an *AE*, the incoming person is inserted into an occupant set. If the incoming individual comes infectious, its transmission within the local population is computed using Equation 1. If it is uninfected, then its within-host progression is computed using the PTTS. The incoming person's departure is scheduled based on the parameter *mean.staytime*. The individual's next departure time is computed, and departure event *DE* is scheduled by the location to itself as reminder to eject the individual

from the location's state at that time. At the time the departure event is processed, a destination location (either local to the region or in another region across processors) is selected, the travel time computed, and the state of the person corresponding to the moment of departure at the source location is packed into the event and sent.

```
Forward Algorithm
If(Event is AE)
  Insert AE.Person into Occupant Set
  Incorporate AE.Person's random seed locally
  dt=Randomized depart time of AE.Person
  Schedule DE for AE.Person at now+dt
  If(AE.Person turned infectious between
      now and its departure time at source)
      Progress infection state of AE.Person
       (schedule SCE for AE.Person)
  End If
  Compute Equation 1 (reaction function)
  Determine new infections due to Equation 1
  For each person P in occ newly infected now
      Progress infection state of P by PTTS
       (retract previous SCE of P and
        schedule a new SCE for P)
  End For
Else If(Event is DE)
  Locate person P in occ with identifier DE.pid
  Remove P from occ
  Determine destination L (diffusion function)
  If P has local pending SCE, note residual □t
  Determine travel time dt (diffusion function)
  Schedule AE for P to L at now+dt
Else /*Event is SCE*/
  Locate person P in occ with ID SCE.pid
  Progress infection state of P by PTTS
    (schedule SCE for P)
  If( P just turned infectious)
    Compute Equation 1 (reaction function)
    Determine new infections due to Eqn 1
    For each person Q in occ newly infected
       Progress infection state of Q by PTTS
       (retract previous SCE of Q and
        schedule a new SCE for Q)
    End For
  End If
End If
```

Figure 4: Forward Event Processing at each LP

The processing of a *DE* marks the exit of that person from its current location. The outgoing individual is removed from the occupant set and an *AE* is scheduled at the destination location.

When an individual *P* within a location changes its state of infection, a *SCE* is triggered. The infection state of *P* is then evolved using its PTTS. If the final state is infectious, then its effect on the rest of the occupants within the location is computed using Equation 1.

A couple of important nuances must be considered for correctness: (a) a person turning infectious while in transit, and (b) random number stream continuity must be maintained across processors when persons cross processors. The first nuance is taken into account by noting the residual time of infection dormancy in *AE* so that the receiving location can verify the special condition and act accordingly (special case trapped in Forward Algorithm by the first conditional in processing *AE*). The latter nuance is handled by shipping in *AE* the random number generator seed of that person along with the person's state. The receiving processor restarts its random number stream accurately at the received starting seed for that person.

3. Reverse Execution Model

We now turn to the reverse execution handlers that make use of the information generated in the modified forward execution to undo forward operation.

3.1. Modified LP Data Structures

Since the original forward-only model is not reversible as is, a few variables need to be added to achieve reversibility. Note that this does not imply the use of state-saving; it only implies the increase of sequential processing memory need by a constant factor. In other words, no state *log* is accumulated as a result of this augmentation, but only the original copy of the (single) state is increased by a *constant* amount.

```
(Added) Person State
prev_istate: Previous Infection State
infect_ts: Infected Time
```

Figure 5: State Added to Person for Reversibility

The variables added to each person are shown in Figure 5. The *prev_istate* is used to mark its previous infection state to remember a irrecoverable previous state, if any, in the PTTS, encountered by the person. The *infect_ts* is added to disambiguate the identity, if any, of event that triggered the infection of this person.

```
(Added) Location Data Structures (LP State)
dep: "Departed" Set of Persons (states)
dmap: Map from ID to Person in dep Set
```

Figure 6: State Added to each Location (LP) for Reversibility

Figure 6 shows the addition of a "departed" set to

each location to accommodate reversibility of *DE*. This is used to "keep the person around" in case the departure event is rolled back. Again, this does not constitute state saving by itself. To compare, even if the best (incremental) state saving techniques are employed, two copies of person's state would be generated in the incremental state log for every move from occupant set to departed set, bringing the total memory to 3 person states, compared to the two states in our reverse computing scheme.

3.2. Modified Forward Event Processing

The forward event processing is augmented as follows: (a) any time an infection is initiated on a person, its infection time is noted in the person's state, (b) departing persons in *DE* are moved to the "departed set" rather than discarded (c) previous state of a person is noted in the corresponding *SCE* event for every state change. Due to space limitations, the modified forward algorithm is not reproduced here.

3.3. Backward Event Processing

The reverse execution by locations is shown in Figure 7.

```
Backward Algorithm
If(Event is AE)
  For each person P in occ infected at now
     Restore its states to previous in PTTS
     Reverse its random number generator
  End For
  Remove AE.Person from occ
Else If(Event is DE)
  Locate person P in dep with identifier DE.pid
  Remove P from dep
  Add P to occ
Else /*Event is SCE*/
  Locate person P in occ with ID SCE.pid
  If( P is infectious)
    For each person Q in occ infected now
       Restore its states to previous in PTTS
       Reverse its random number generator
    End For
  End If
  Restore state of P to previous in PTTS
  Reverse its random number generator
End If
```

Figure 7: Backward Event Processing at each LP

The reversal of arrival event initiates the reversal of locally scheduled departure events, and undoes local infection state change event for the subject individual.

The departure set is maintained should a *DE* need to be reversed in the future. This is used to restore an optimistically departed person back into the occupant set.

Since, as part of processing an *SCE*, the previous infection state is stored, it is restored in backward execution. Same holds true for detecting the set of infected events when the arrival of an infectious person and/or a state change event causes one or more infections. The set of infected individuals is detected accurately by comparing their infection timestamps with the current simulation time.

3.4. Event Commit Operations

An important fallout of the use of a departure set is that the size of the set accumulates over simulation time, since all departed persons are tentatively retained there. Thus, they need to be reclaimed. This is easily done by periodically flushing those entries that contain persons whose departure time is less than the global virtual time. Thus, an additional conditional is added to the event processing loop that detects this condition and the departure set is periodically flushed.

3.5. Random Number Generation

We use the well-known technique of reversible random number generation [19] to undo the several random number generation calls that appear in the model. These appear in the state transition function for PTTS, the computation of infection probability in the reaction function, and the selection of time and destination in the mobility determination (diffusion).

4. Performance Study

We now present a performance study of an implementation of this reversible model.

4.1. Implementation

The application was developed in C++ on top of a simulation engine that supports the concepts of logical processes (LPs), events for exchanging time-stamped messages among logical processes and virtual time-synchronized delivery of events to LPs. The engine avoids all use of collective communication calls and implements a variant of global virtual time synchronization that is purely asynchronous in nature. It avoids blocking in all places and is built with scalable reduction algorithms that are realized in user-space using point-to-point, non-blocking communications.

4.2. Hardware and Software Platforms

The simulations were run on Cray XT4, Cray XT5 and Blue Gene/P (BG/P) machines at the National Center for Computational Sciences (www.nccs.gov).

The XT4 contains 7,832 compute nodes, each node containing a quad-core 2.1 GHz AMD Opteron processor with 8 GB of memory. The nodes are connected via a high-bandwidth SeaStar interconnect. Internally, the MPI implementation is based on Cray's implementation of Portals 3.3 messaging interface. At the time when experiments were run on the XT5, it contained 36,864 quad core AMD Opteron Budapest processors with 2GB of memory per core. The nodes are connected bu a SeaStar-II interconnect. The Blue Gene/P that was used is a 27 TF system consisting of 2048 850 MHz IBM quad core 450d PowerPC processors and 2GB of memory per node.

4.3. Benchmark Description

The performance of the application is studied with respect to changes in: (a) population distribution (b) reaction-diffusion parameters and (c) mobility parameters. Several combinations of the above model parameters were tested. However, due to limitations of space, we present only a few, which we describe next. Two different population distributions, labeled by **I** and **II**, were explored. For the same total population, the number of individuals per location (LP) in distribution **I** is one order of magnitude higher than in distribution **II**. In addition, two different assignments of the reaction-diffusion and mobility parameters, labeled **a** and **b**, were chosen in combination with the above population distributions. Their qualitative descriptions are provided in Table 2 with exact values in Figure 8.

Table 2: Qualitative description of the sets of parameters chosen for our experiments.

	a	**b**
Reaction	Level I	Level II
Diffusion	Infectivity is greater than susceptibility	Susceptibility is greater than infectivity
Mobility	High	Low

reaction	**a**	**b**
normal-UNINFECTED-LATENT	0.9	0.5
normal-LATENT-INFECTIOUS	1	1
normal-INFECTIOUS-RECOVERED	1	1
normal-UNINFECTED-INCUBATING	0.1	0.5
normal-INCUBATING-ASYMPT	1	1
normal-ASYMPT-RECOVERED	1	1
normal-RECOVERED-RECOVERED	1	1
vaccinated-UNINFECTED-RECOVERED	0.5	0.5
vaccinated-UNINFECTED-LATENT	0.2	0.3
vaccinated-LATENT-INFECTIOUS	1	1
vaccinated-INFECTIOUS-RECOVERED	1	1
vaccinated-UNINFECTED-INCUBATING	0.3	0.2
vaccinated-INCUBATING-ASYMPT	1	1
vaccinated-ASYMPT-RECOVERED	1	1
vaccinated-RECOVERED-RECOVERED	1	1

diffusion	**a**	**b**
infectivity	1	0.5
susceptibility	0.5	1
transmissibility	0.0005	0.0005

mobility	**a**	**b**
locality (%)	75	90
lookahead (hr)	0.3	0.5
mean_stay_time (hr)	8	5
mean_local_travel_time (hr)	0.5	1
mean_remote_travel_time (hr)	8	12

Figure 8: Parameter sets used for simulations.

4.4. Experiments

Several simulations were run based on combinations of the above parameter sets. Here, we present those that expose key performance aspects. The legends in the graphs indicate the combination of **[reaction diffusion mobility]** parameter sets used for those runs.

One of the effects of population distribution on the parallel performance can be observed in Figure 9 which shows the relative speedup for population sets I (top) and II (bottom) on processor counts varying from 128 to 4096 on the BG/P architecture. The reaction-diffusion and mobility parameters for both are identical as are the total populations. In this model, the probability that an individual chooses a destination location that is resident on a remote region is the same as that of choosing one that is locally available, As a result, region boundaries are crossed more often when there are less destination locations available locally. Since regions are mapped to processors, the resulting communication overhead increases as the number of locally available locations decreases. This effect is evident in Figure 9 which shows the degradation of parallel performance in going from 10 to 100 locations per region for the same total population.

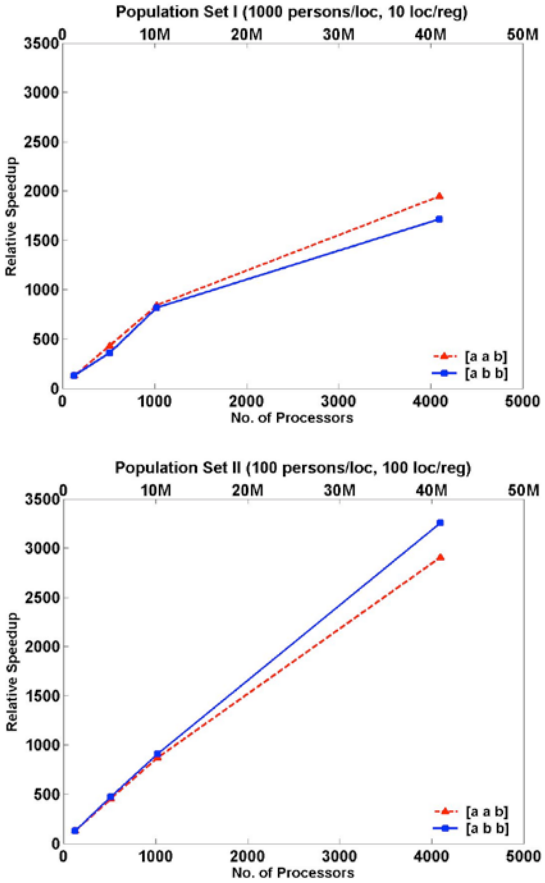

Figure 9: Speedup on BG/P for two different population distributions with the same total.

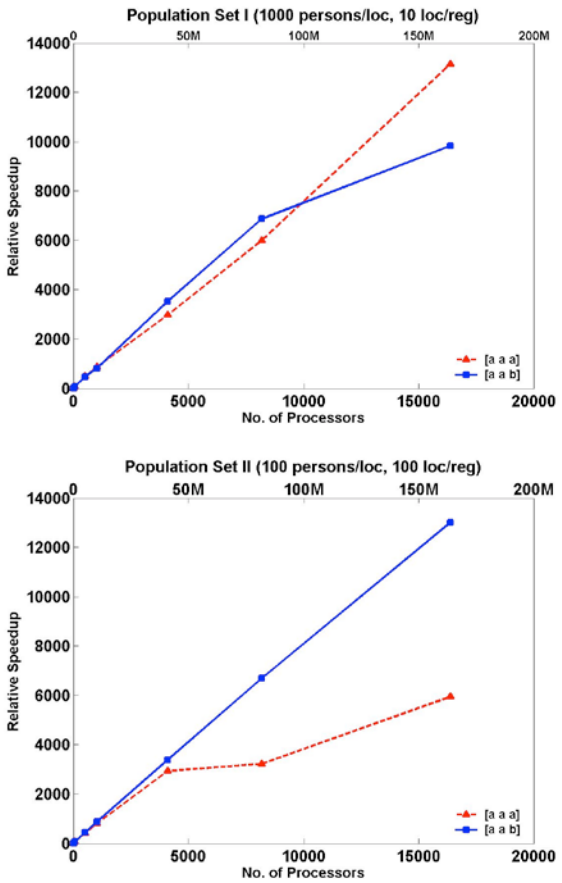

Figure 10: Speedup on Cray XT4 for two different population distributions with the same total.

Similar experiments were carried out on the Cray XT4 with simulation runs carried out on up to 16,384 processors. In Figure 9, only the diffusion parameter set is varied while in Figure 10 only the mobility parameter set is varied. For the same parameter set **[a a b]**, the parallel performance is seen to improve in Figure 10 with an increase in the number of locations that are locally available, as explained above. This is not the case, however, for the scenario with parameter set **[a a a]** (see Figure 10). This highlights the competing effects of mobility of individuals within a population and the distribution of the individuals within it.

Figure 11 indicates that for a wide variety of reaction-diffusion and mobility conditions, the rollback efficiency of the optimistic parallel simulation remains under control (well under 10% for all tested scenarios). Finally, Figure 12 demonstrates the scalability of the simulation to 65,536 processors on a Cray XT5 platform with 10,000× speedup.

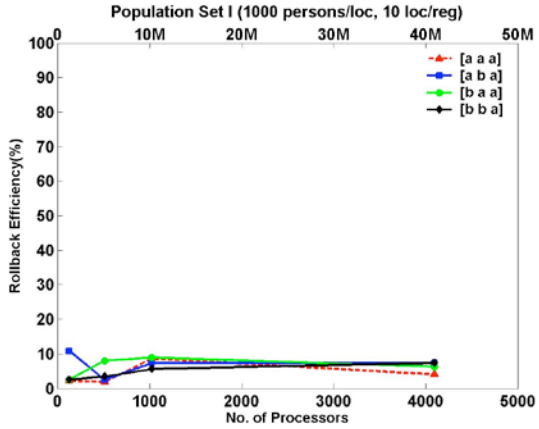

Figure 11: Rollback Efficiency on BG/P.

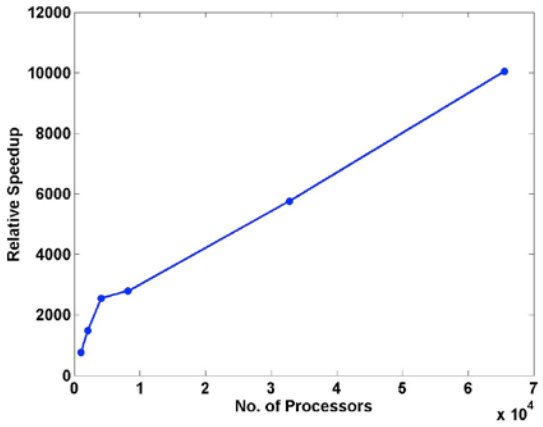

Figure 12: Speedup on Cray XT5.

5. Summary

A discrete event formulation, its reverse-computation-based parallel execution, and a performance study of an actual implementation have been presented here. The focus of this article was more on the computational aspects, as opposed to the use of the simulation for domain science in epidemic simulations. Scalability of the overall system to very large parallel computing platforms has been demonstrated, with the largest being executed on 65,536 processor cores. The net outcome of this work is the clear demonstration that the PDES technology has now reached a point at which very large epidemic simulations can now be realized with significant speedup (over $10^4\times$) using large parallel computing platforms.

6. Ongoing Work

We are currently incorporating additional modeling concepts, namely, resource constraints, and critical thresholds, whose significance was recently suggested [20]. Also being incorporated are prevention, mitigation and intervention mechanisms and their effects. We are also investigating the performance differences between optimistic *vs*. conservative execution for the epidemic models. Similar performance comparison of interest is between copy state saving and incremental state saving in comparison to the reverse computation reported here. We are also pursuing the use of actual interconnectivity graphs (e.g., social and transportation networks) for diffusion.

Acknowledgements

This paper has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy. Accordingly, the United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

References

[1] M. Z. Gojovic, *et al.*, "Modelling Mitigation Strategies for Pandemic (H1N1) 2009," *CMAJ,* vol. 181, pp. 673-680, 2009.

[2] M. Barthelemy, *et al.*, "Velocity and Hierarchical Spread of Epidemic Outbreaks in Scale-Free Networks," *Phys. Rev. Lett.,* vol. 92, p. 178701, 2004.

[3] R. Pastor-Satorras and A. Vespignani, "Epidemic Dynamics in Finite Size Scale-free Networks," *Phys. Rev. E,* vol. 65, p. 035108, 2002.

[4] R. Pastor-Satorras and A. Vespignani, "Epidemic Spreading in Scale-Free Networks," *Phys. Rev. Lett.,* vol. 86, pp. 3200-3203, 2001.

[5] D. J. Watts and S. H. Strogatz, "Collective Dynamics of 'Small-world' Networks," *Nature,* vol. 393, pp. 440-442, 1998.

[6] C.-Y. Huang, *et al.*, "Influences of Resource Limitations and Transmission Costs on Epidemic Simulations and Critical Thresholds in Scale-Free Networks," *SIMULATION,* vol. 85, pp. 205-219, March 1, 2009 2009.

[7] J. Epstein, "Modelling to Contain Pandemics," *Nature,* vol. 460, p. 687, 2009.

[8] G. V. Bobashev, *et al.*, "A Hybrid Epidemic Model: Combining The Advantages of Agent-based and Equation-based Approaches," in *Proc. of the Winter Simulation Conference*, ed, 2007.

[9] S. Riley, "Large-scale Spatial-Transmission Models of Infectious Disease," *Science,* vol. 316, pp. 1298-1301, 2007.

[10] S. Eubank, *et al.*, "Modeling Disease Outbreaks in Realistic Urban Social Networks," *Nature,,* vol. 429, pp. 180-184, 2004.

[11] M. E. Halloran, *et al.*, "Modeling Targeted Layered Containment of an Influenza Pandemic in the United States," *PNAS,* vol. 105, pp. 4639-4644, 2008.

[12] T. C. Germann, *et al.*, "Mitigation strategies for pandemic influenza in the United States," *PNAS,* vol. 103, pp. 5935-5940, April 11, 2006 2006.

[13] J. Parker, "A Flexible, Large-scale, Distributed Agent-based Epidemic Model," presented at the Winter Simulation Conference, Piscataway, NJ, 2007.

[14] J. M. Linebarger, *et al.*, "Smallpox over San Diego: Joint Real-Time Federations of Distributed Simulations and Simulation Users under a Common Scenario," presented at the Proceedings of the 21st International Workshop on Principles of Advanced and Distributed Simulation, 2007.

[15] D. W. Bauer and M. Mohtashemi, "An application of parallel Monte Carlo modeling for real-time disease surveillance," in *Simulation Conference, 2008. WSC 2008. Winter*, 2008, pp. 1029-1037.

[16] C. L. Barrett, *et al.*, "EpiSimdemics: An Efficient Algorithm for Simulating the Spread of Infectious Disease over Large Realistic Social Networks," presented at the Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, Austin, Texas, 2008.

[17] K. R. Bisset, *et al.*, "EpiFast: A Fast Algorithm for Large Scale Realistic Epidemic Simulations on Distributed Memory

Systems," in *Proc. of Int'l. Conf. of Supercomputing*, ed, 2009, pp. 430-439.

[18] D. W. Bauer*, et al.*, "Scalable Time Warp on Blue Gene Supercomputers," presented at the Proceedings of the 2009 ACM/IEEE/SCS 23rd Workshop on Principles of Advanced and Distributed Simulation, 2009.

[19] C. Carothers*, et al.*, "Efficient Optimistic Parallel Simulations using Reverse Computation," *ACM Transactions on Modeling and Computer Simulation,* vol. 9, pp. 224-253, 1999/07/01 1999.

[20] C. Y. Huang*, et al.*, "Influences of Resource Limitations and Transmission Costs on Epidemic Simulations and Critical Thresholds in Scale-Free Networks," *Simulation,* vol. 85, pp. 205-219, 2009.