Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

PARALLEL SIMULATION OF THE GLOBAL EPIDEMIOLOGY OF AVIAN INFLUENZA

Dhananjai M. Rao

CSA Department
Miami University
Oxford, OH 45040. U.S.A.

Alexander Chernyakhovsky

William Mason High School
6100 Mason-Montgomery Rd.
Mason, OH, 45040. U.S.A

ABSTRACT

SEARUMS is an Eco-modeling, bio-simulation, and analysis environment to study the global epidemiology of Avian Influenza. Originally developed in Java, SEARUMS enables comprehensive epidemiological analysis, forecast epicenters, and time lines of epidemics for prophylaxis; thereby mitigating disease outbreaks. However, SEARUMS-based simulations were time consuming due to the size and complexity of the models. In an endeavor to reduce time for simulation, we have redesigned the infrastructure of SEARUMS to operate as a Time Warp synchronized, parallel and distributed simulation. This paper presents our parallelization efforts along with empirical evaluation of various design alternatives that were explored to identify the ideal parallel simulation configuration. Our experiments indicate that the redesigned environment called SEARUMS++ achieves good scalability and performance, thus meeting a mission-critical objective.

1 INTRODUCTION

Avian influenza is a viral disease caused by H5N1, a highly virulent strain of the influenza-A virus, that has the potential to cause the next global pandemic (CDC 2006, WHO 2006). Infected migrating waterfowl, in which the virus is endemic, are the primary vectors for causing intercontinental spread of the disease (Normile 2006). The virus rapidly spreads from waterfowl to poultry and humans through contaminated water, feed, and surfaces. It has devastating impacts on poultry farming and has resulted in numerous human deaths (CDC 2006). Targeted antiviral prophylaxis is the prevention and control mechanism for influenza. Unfortunately, the constantly changing antigenic characteristics of H5N1 reduces efficacy of vaccinations (WHO 2006). Furthermore a myriad of technological issues pose serious hurdles to manufacturing and distribution of even small volumes of H5N1 vaccines (WHO 2006).

Consequently, strategies for containing epidemics and mitigating pandemics are of vital importance (Ferguson et al. 2006, Longini et al. 2005). Such strategies require extensive knowledge about its global epidemiology, epicenters of disease outbreaks, and timelines for targeted prophylaxis using low efficacy vaccinations. An effective methodology for discovering epidemiological knowledge is simulation-based analysis using a descriptive, ecological model (Ferguson et al. 2006, Longini et al. 2005). Accordingly, we have developed a modeling, simulation, and analysis environment called SEARUMS. It is a Java-based graphical modeling, simulation, and analysis environment that is specialized for epidemiological study of avian influenza.

In SEARUMS, simulation-based analyses are conducted using an individual, agent-based, spatially-explicit model of global epidemiology of avian influenza. Agents essentially implement the classical SIR (Susceptible-Infected-Removed) mathematical models (Anderson and May 1992) used to describe the epidemiological behaviors of the the three salient entities, namely: waterfowl, poultry, and humans. The agents are specifically designed to ease effective use of real world statistical data. The conceptual design of each agent is based on discrete time Markov processes (Rao et al. 2007). Figure 1 presents an overview of the three Markov processes along with the SIR mathematical model.

The global epidemiology of avian influenza is modeled using a collection of the aforementioned agents. The models, called Eco-descriptions, have been developed using real-world statistical data on: waterfowl migration (GROMS 2006), waterfowl species that are at higher risk to carry the virus (Brahmbhatt 2006), global poultry population and distribution (GLiPHA 2007), and global human population distribution information (SEDAC 2007). The models have been calibrated, validated, and verified through extensive simulations. Verification and validation was performed by confirming that the timing and chronology of several outbreaks observed in the simulations correlate with several significant real-world incidents reported by the World

978-1-4244-2708-6/08/$25.00 ©2008 IEEE

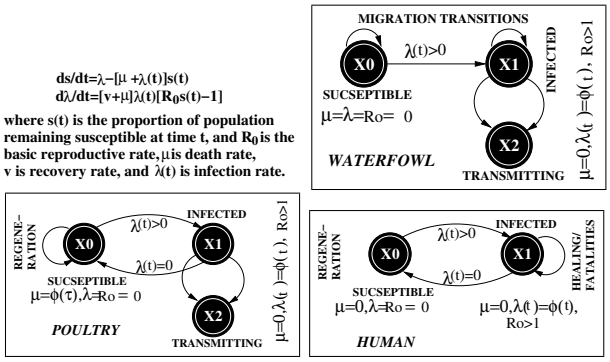


Figure 1: Overview of SIR model based Markov Processes.

Health Organization (WHO) (WHO 2006). The aforementioned investigations have already been reported and are not the focus of this paper. Consequently, we have provided a succinct overview while providing details in an on-line appendix (SEARUMS 2008). Further details on SEARUMS, the Eco-descriptions, their verification and validation along with inferences drawn from several epidemiological studies are available in the literature (Rao et al. 2007).

2 MOTIVATION

The earlier implementation of SEARUMS used a Java-based, multi-threaded discrete event kernel for simulation. The simulation kernel was designed to operate on shared-memory (multi-core or multi-processor) systems. It used a centralized event queue with minimal locking for scheduling and processing timestamped events in correct causal order. Multiple threads operate in a synchronous manner, processing concurrent events scheduled at a given simulation time. A more detailed description of the simulation kernel is available in the literature (Rao et al. 2007).

The Java-based simulation back-end of SEARUMS performed well for small models, involving less than 1000 agents (Rao et al. 2007). However, as the size of the model increased, the simulation kernel did not scale effectively. Consequently, simulating medium sized models (with 5000 to 8000 agents) required more than 2 to 3 hours per run. Hundreds of such runs were necessary for thorough epidemiological analysis; making such studies prohibitively time consuming. Furthermore, the supercomputing cluster available for our research had a standard distributed memory architecture. Consequently, the shared-memory design could not effectively utilize the computational infrastructure beyond a single node. Consequently, we endeavoured to redesign SEARUMS' simulation back-end to address these issues.

3 OVERVIEW OF PROPOSED SOLUTION

In order to circumvent the issues presented in Section 2, we redesigned our simulation infrastructure to execute as an optimistic parallel and distributed simulation. Parallel simulation is a natural candidate to enable effective use of supercomputing clusters built using distributed memory architectures (Deelman and Szymanski 2002, Fujimoto 1990, Glass et al. 1997, Maniatty et al. 1999). However, ensuring scalability required that the latent parallelism available in the model is effectively utilized. Accordingly, to maximize parallelism we proposed to utilize Time Warp, the popular optimistic synchronization methodology (Fujimoto 1990). Time Warp was also chosen based on our prior experiences and due to encouraging parallelization results reported by other investigators for similar spatially-explicit models (see Section 4).

Accordingly, we redesigned SEARUMS' simulation infrastructure in C++ using a general purpose simulation framework called WESE (Rao 2003). WESE was chosen as the target parallel simulation framework for several reasons. WESE is a general purpose, web-enabled, Time Warp synchronized framework that eases development of parallel and distributed simulations (Rao 2003). It has been developed by suitably extending the WARPED simulation kernel. WARPED provides the core Time Warp related infrastructure that WESE further customizes to provide additional features, including a web-based interface. The web interface supports a text-based protocol that can be used to establish, initiate, monitor, and control parallel simulations. Moreover, WESE manages the task of centralizing standard output from various computational nodes to ease visualization. This feature facilitated seamless integration with SEARUMS' Java-based modeling, visualization, and analysis GUI; thereby eliminating the need to re-implement the fully functional and comprehensive frontend. The aforementioned aspects make WESE an ideal framework for developing our redesigned environment called SEARUMS++. An architectural overview of SEARUMS++ along with details on the redesigned agents is presented in Section 5.

4 RELATED RESEARCH

Having presented an overview of the proposed approach we proceed to compare and contrast our research with earlier investigations. Due to space restrictions, we discuss only the very closely related research in this section. Readers are referred to the references and literature for a more comprehensive survey of related works (Rao et al. 2007). Optimistic parallel simulation of epidemiology of avian influenza is a unique application. Nevertheless, the models fall under the category of spatially-explicit models and several researchers have investigated the application of parallel simulation methodology to such models.

Deelman and Szymanski discuss the use of Time Warp for simulating spatially-explicit models of Lyme disease on an IBM SP, based on a shared memory architecture (Deelman and Szymanski 2002). They use a Breadth-First Rollback (BFR) method that utilizes incremental state saving to rapidly recover from causal violations. They report almost linear speedup for models of different sizes. Maniatty *et al* illustrate the use of parallel simulations to simulate generic evolutionary, spatially explicit models using genetic algorithms and cellular automata (Maniatty et al. 1999). They present stepwise refinement of three systems called TEMPEST, STORM, and GALE to yield the desired models and MPI-based parallel simulation. Their experiments utilized both conservative and optimistic synchronization methods. From their empirical analysis on a shared memory SGI Origin 2000, they conclude that for spatially explicit models with large diversity of time scales, an optimistic approach is more efficient than the conservative one (Maniatty et al. 1999).

Glass et al. (1997) discuss the use of Time Warp synchronized parallel simulations for analyzing spatially explicit, ecological models on shared memory multi-processor architectures. They propose the use of shared object states to circumvent some of the bottlenecks reported by Deelman. Eubank (2002) presents MPI-based parallel simulation of spread of disease among individuals in a large urban population over the course of several weeks. The simulations were designed to run on a distributed memory platform using a conservative synchronization protocol.

In contrast to the research conducted by Deelman (Deelman and Szymanski 2002), Maniatty (Maniatty et al. 1999), and Glass (Glass et al. 1997), our research focuses on parallel simulation on a distributed memory cluster rather than a shared memory architecture. Consequently, our design approaches and trade-offs are different from theirs. However, similar to these three investigations we also use the Time Warp synchronization protocol. The use of Time Warp distinguishes our research from that of Eubank (Eubank 2002) who uses conservative synchronization.

From an epidemiological perspective, our investigations assume and reflect the current, real-world situation; *i.e.,* H5N1 is yet to mutate into its pandemic state and human-to-human transmission is unsustained. Furthermore, our research aims to utilize waterfowl migration data to forecast epicenters and timelines of potential epidemics as well as study effects of targeted, proactive prophylaxis to mitigate a pandemic. In addition, we also emphasize socio-economic impacts on global poultry farming which is currently experiencing significant economic impacts. The aforementioned aspects notably distinguish our efforts from several recent investigations reported by Ferguson et al. (2006), Longini et al. (2005), and LANL (2006). The latter investigations are based on the premise that H5N1 has already mutated to a pandemic form and epidemics are being caused primarily due to human-to-human transmission. Such a scenario continues to remain only a possibility at the time of this writing. On the other hand, similar to the latter three investigations, we also use the classical SIR (Susceptible-Infected-Removed) mathematical approach for modeling.

5 SEARUMS++

An architectural overview of the redesigned, hybrid (combination of Java and C++) system called SEARUMS++ is shown in Figure 2. It has been developed by interfacing SEARUMS with a Time Warp synchronized simulation back-end developed using WESE. All user interactions with SEARUMS++ are preformed via the graphical `model editor` to create a model called an Eco-description. The `model editor` interacts with the `visualization subsystem` to provide geographic (using multi-zoom maps) and statistical (using a variety of configurable charts) representations. The `model editor` essentially stores configuration information in *skeleton* agents instantiated from the `agent repository` using reflection. The skeleton agents were introduced into the `agent repository` as a part of this research. The skeleton agents serve as place holders to facilitate user interactions, GUI development, and persistence. The actual agents that perform the core simulation-time activities are part of WESE factories that are deployed on various compute nodes used for parallel simulation.

A WESE `Factory` acts as an agent repository and a simulation server. It has been developed in C++ using WESE's Application Program Interface (API). Figure 2 illustrates the modules constituting a WESE `Factory`. The `communication subsystem` handles the tasks of interacting with remote SEARUMS++ clients and other WESE factories used for simulation. The `gateway` modules uses the `communication subsystem` to provide the initial entry point to a WESE `Factory`. It creates a new `session manager` for each unique session initiated with the factory. The `session manager` performs the task of interacting with the `agent factory` to create the actual agents and locally establishes part of the Time Warp synchronized parallel simulation.

The `session manager` then initiates a distributed simulation by interacting with `session managers` on other WESE `Factories` used in a simulation. Agents in a single WESE `Factory` are part of the same process and are scheduled using a single thread of execution. Scheduling of events is performed using a shared Least-Time-Stamp-First (LTSF) event queue. Therefore, events exchanged between agents on the same WESE `Factory` will never cause rollbacks. However, the agents are not coerced into synchronizing with each other. Conversely, inter-`Factory` events that are exchanged over the network give raise to straggler events resulting in rollbacks. The core Time Warp

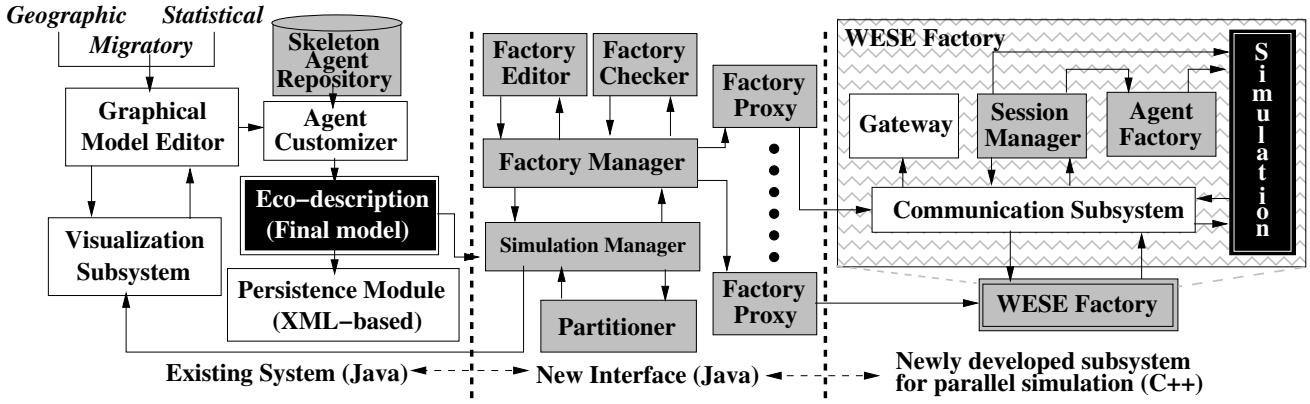


Figure 2: Architectural overview of SEARUMS++. Modules in gray have been newly developed as a part of this research.

synchronization feature of WESE has been realized by using a modified version of the WARPED simulation kernel (Rao 2003). WARPED also handles Global Virtual Time (GVT) based garbage collection and generation of optimistic I/O. Each `session manager` also performs the task of redirecting standard output from local simulations back to a SEARUMS++ client for visualization. A more detailed description of WESE and the process of developing a WESE `Factory` are available in the literature (Rao 2003).

The graphical front-end and the WESE-based parallel simulation back-end have been coupled together using a Java-based interface layer. This interface, shown in Figure 2, has been developed as a part of this research. The `simulation manager` is the core module that establishes, controls, and coordinates all the simulation related activities. It first interacts with the `partitioner` to suitably partition the Eco-description for parallel simulation. Then it uses the list of factories configured in the `factory manager` to interact with various factories to perform a parallel simulation. Protocol details involved in communicating with a WESE `Factory` are handled by individual `factory proxy` modules. The `factory manager` coordinates the various `factory proxy` modules and eases interactions between them and the `simulation manager`.

The `simulation manager` also performs the task of routing application-level messages to suitable visualization modules that update corresponding skeleton agents. The skeleton agents provide a dynamic (*i.e.,* during simulation), intuitive, graphical representation of the actual agents on various factories. The agents and their behaviors reflect their conceptual models shown in Figure 1 and are briefly described below:

Waterfowl Group Agent (WGA): A `WGA` represents a flock of waterfowl. The agent starts at a given coordinate (represented by latitude and longitude) and periodically (every 1 hour of simulation time) changes its position along a predefined migratory path. The migration pathways and migratory patterns have been developed from data published by various international organizations (GROMS 2006, Hagemeijer and Mundkur 2006). Currently, `WGA` instances reflect high risk waterfowl species in which H5N1 is endemic. Hence these agents are initialized to carry some infection. This agent provides different configurations to model intra-flock and inter-flock infection spread (Rao et al. 2007). Inter-flock infections occur when the agent migrates and comes in contact with any other agent in the Eco-description. In order to correctly model infection spread, the state of this agent is large and contains the following information: current coordinate (latitude and longitude), current migration point (the per agent migration path is fixed during modeling and is not in the state), flock population, infection percentage, a blacklist of recently infected agents who should not be reinfected for a period of 168 hours, and a list to track changes in overlap areas with neighboring agents.

Poultry Group Agent (PGA): A `PGA` models a non-migrating flock of poultry. The activities of this agent commence when it is infected by another agent in the simulation. Infections trigger intra-flock spread, inter-flock spread, and culling-regeneration processes in a `PGA`. Various behaviors, including inter-flock and intra-flock infection spread models, can be suitably configured during model development. The state of this agent is similar to that of the `WGA` except it does not contain migration data but has state information to model culling and regeneration processes.

Human Group Agent (HGA): The `HGA` is used to model a group of humans living in proximity of each other with a consistent population density. Similar to the `PGA`, an infection triggers various activities of this agent. Currently, human-to-human spread of avian influenza has been rare and unsustained. Consequently, this agent only performs healing and death processes. Accordingly, its state is small and contains the following data: number of infected humans, number of humans healing, and count of human fatalities.

Each of the aforementioned agent has been developed as classical Time Warp Logical Process (LP). The temporal behaviors and life cycle processes of agents are coordinated by scheduling suitable timestamped events. The space occupied by each agent is approximated to circular regions with even density. Such a modeling approach is commonly used in spatially explicit ecological models (Deelman and Szymanski 2002, Ferguson et al. 2006, Glass et al. 1997, Longini et al. 2005, Maniatty et al. 1999). The spatial interactions between agents are modeled using `EcoArea` components that represents Earth's surface. `EcoArea` components receive update events from an agent initially and whenever it changes its attributes, such as: current coordinate, infection percentages, and population changes. The `EcoArea` components tracks agents in its purview by maintaining a list in its state. It uses the information to detect and trigger interactions between overlapping agents by scheduling timestamped events.

SEARUMS++ provides two different `EcoArea` components that can be used in the three different configurations enumerated below. The three configurations were developed in an incremental manner to address scalability and performance issues observed in the previous ones. However, note that immaterial of the configuration used, the final application-level events are identical in all cases. The three different configurations explored in the design of SEARUMS++ were:

Config #1: Unified EcoArea: This component represents the complete surface of the earth as shown in Figure 3(a). One unique component is instantiated on each compute node used for parallel simulation by the `simulation manager`. Agents partitioned to that node only send `update` events to their their local `Unified EcoArea`. Conversely, each `Unified EcoArea` component aggregates its local changes and broadcast updates to other `EcoAreas` to ensure coherence of agent information. These broadcasts may cause additional local `proximity` events to be generated. Agents receive `proximity` notifications only from their local `EcoArea`. Agents utilize the information in `proximity` events generated by `EcoArea` components and directly schedule `infection` events to overlapping agents. The objective of this design was to minimize inter-`Factory` communication that ultimately plays a crucial role in causing rollbacks. However, as presented Section 6, this configuration did not yield effective performance. Therefore, we explored the design alternatives discussed below to circumvent the bottlenecks observed in this configuration.

Config #2: Split EcoArea: This component represents a specific rectangular region of the Earth's surface. Areas represented by `Split EcoArea` components are distinct and do not overlap. One unique `Split EcoArea` component is created on each WESE `Factory`. Currently, a minimum of 2 factories are necessary to simulate using this configuration. Furthermore, the `partitioner` (see Figure 2) distributes agents based on their geographical coordinates to the corresponding `Factory`. When `Split EcoArea` is used in a simulation, agents communicate only with the `EcoAreas` they overlap. Figure 3(b) illustrates this configuration involving four factories. Note that as agents move they appropriately register and unregister themselves from the corresponding `EcoAreas`. However, in this mode the agent LPs do not *physically* migrate from one WESE factory to another. Instead, inter-`Factory` interactions occur over the network which resulted in increased rollbacks as the number of factories was increased. This resulted in diminished scalability as discussed in Section 6. Consequently, we further enhanced the simulation by introducing *physical* migration of agents as explained below.

Config #3: Split EcoArea with Proxies: This configuration builds on the previous configuration by introducing the concept of *proxy* agents for all mobile `waterfowl` agents in the simulation. The proxy agents are used to achieve *physical* migration of LPs from one WESE `Factory` to another, tracking the migration pattern of agents. The overall objective is to minimize inter-factory messages (that are exchanged over the network) as they are the primary source of rollbacks in WESE-based simulations. The proxy agent feature has been implemented in the following manner. Initially, one deactivated proxy agent is created by the `simulation manager` on each WESE `Factory` for every mobile agent in the model. Deactivated agents do not perform any operations and remain dormant in the simulation. Based on the initial geographical location of an agent, the appropriate proxy is activated and it performs the normal operations of a `waterfowl` agent. When the agent migrates across its local `EcoArea`, it first deactivates itself. Then it schedules an activation event to the appropriate proxy object on a different WESE `Factory`. Furthermore, dormant proxies simply forward events (as they may already be scheduled) to the active proxy. Activation, deactivation, and forwarding of events are performed using sub-simulation cycles. Consequently, the proxy configuration was achieved with minimal increase in state size. The `simulation manager` (see Figure 2) performs the task of collating and distributing proxy information during agent creation. This configuration provided the best scalability and performance as described in Section 6.

6 EXPERIMENTS

The experiments conducted to evaluate the effectiveness of the various simulation configuration of SEARUMS++ were performed using three models. Table 1 shows the number of agents constituting each models. All the models include only the high risk species of waterfowl that play a dominant role in intercontinental spread of avian influenza (Hagemeijer and Mundkur 2006, SEARUMS 2008). This subset of

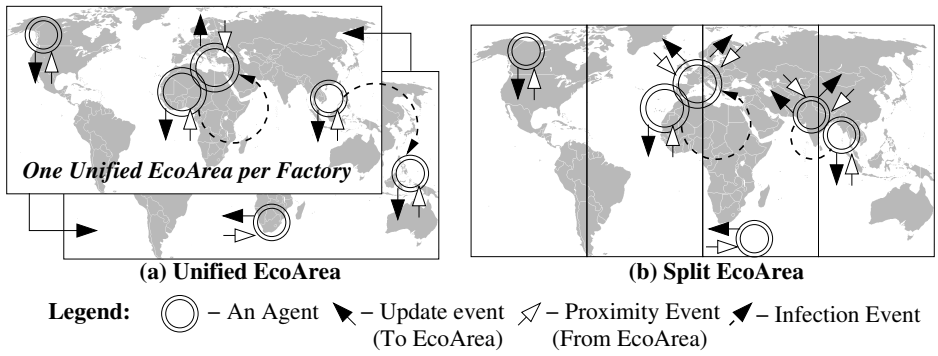


Figure 3: Different simulation configurations involving `Unified` and `Split EcoArea` components.

waterfowl have shown to be sufficient to reproduce the major real-world disease outbreaks as reported by WHO (Rao et al. 2007, WHO 2006). As shown in Table 1, the models have a varying number of humans and poultry agents at different scales of detail (or resolution). Agent data at different scales were generated from poultry and human census data published by GLiPHA 2007 and SEDAC 2007 respectively. All the experiments were conducted on a distributed memory super computing cluster running Linux. Each compute node had four dual-core AMD Opterons^{TM} with 16 GB of RAM. The nodes were interconnected using Gigabit Ethernet. Note that the experiments were performed using the headless mode supported by SEARUMS++ to avoid GUI overheads. Furthermore, the front-end was executed on a separate compute node to eliminate resource contentions with WESE factories.

The simulations were verified by ensuring that the events generated by the parallel simulations were identical to the corresponding events generated by the earlier SEARUMS (Java-based) simulations. Note that the earlier Java simulations have been thoroughly verified and validated by comparing epicenters and time lines of simulated outbreaks against real world outbreaks reported by WHO (Rao et al. 2007). Readers are referred to the references for further details (SEARUMS 2008).

The objective of the first phase of experimentation was to identify the optimal simulation configuration from the three enumerated in Section 5. We used the model `M1` for these experiments. This model was used because it had short run-times facilitating numerous repeated runs. Since the model is small, it has limited concurrency and parallelism (Rao, Chernyakhovsky, and Rao 2007). Therefore it serves as a good stress test to validate the scalability and performance achieved in each configuration. For all of the `Split EcoArea` configurations, the surface of the earth was divided into vertical strips as shown in Figure 3(b). This strategy provided us the best load balance for the various configurations.

Table 1: Characteristics of models used in experiments (Pop means total population represented by agents in scientific notation. See appendix for detailed breakdown)

ID	Number of Agents			Total
	WGA	PGA	HGA	
M1	44	1314	1314	2672
Pop:	4.371e6	1.81e10	6.65e9	2.48e10
M2	44	4251	2160	6455
Pop:	4.371e6	1.81e10	6.65e9	2.48e10
M3	44	5586	5270	10900
Pop:	4.371e6	1.81e10	6.65e9	2.48e10

Prior to the vertical partitioning, we tried dividing the surface into a 2-dimensional grid. However, the grid partitioning scheme resulted in diminished scalability and performance. The reason is that the southern hemisphere does not have sufficient land surface with humans and poultry. Consequently, grid areas in the southern hemisphere did not effectively utilize the computational resources assigned to them. However, partitioning as vertical strips covered both hemispheres providing a better load balance. Currently, we have not explored the option of varying the width of the strips during partitioning.

The graph in Figure 4 shows the simulation execution times (average of several simulation runs) observed for the three configurations. The experiments were stopped when the run times started to steadily increase. As illustrated by the graphs, `Config` #1, namely the `Unified EcoArea`, did not perform well. The root cause of the bottleneck was the significant increase in broadcast messages between `EcoAreas` in this configuration. The increase in inter-factory messages is evident from the corresponding curve in Figure 5(c). As the number of nodes used for parallel simulation were increased, the number of events exchanged

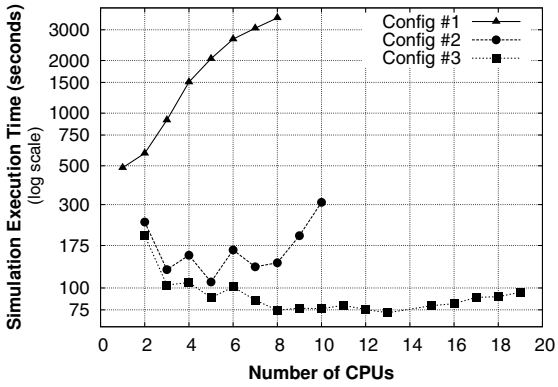


Figure 4: Simulation execution timings for model `M1`

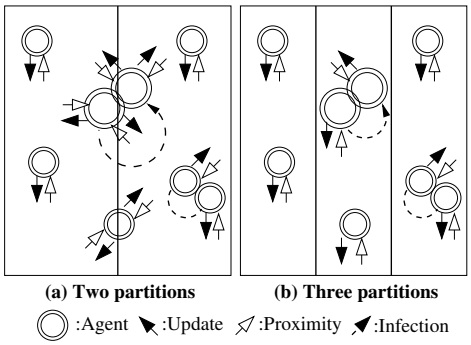


Figure 6: Effect of partitioning on events

between `Unified EcoAreas` grew in a linear fashion. The increase in events causes the number of rollbacks to grow as shown in Figure 5(b), thus degrading the performance.

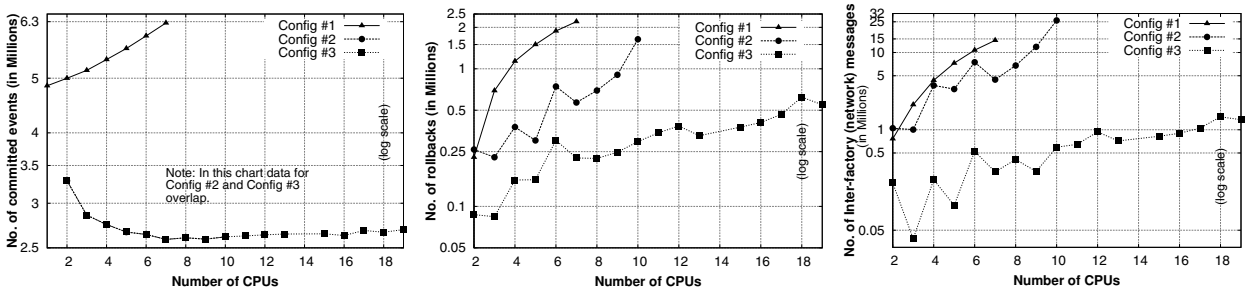
On the other hand, the other two configurations do not involve broadcasting of events between `EcoAreas`. Instead, as described in Section 5, the agents directly schedule events to appropriate `Split EcoAreas` depending on their current geographic location. This feature significantly restricts the necessary cross `EcoArea` interactions. Therefore, as shown in Figure 5(a), the total number of committed events initially decreases as the number of partitions increases. Figure 6 illustrates a simple scenario to assist in reasoning about the decrease in events. As shown in Figure 6(a), in the two partition case the agents overlapping two partitions need to generate and process two sets of events. In contrast, the the three partition case shown in Figure 6(b), one set of events are eliminated; thereby reducing the total number of committed events. However, note that application-level events remain unchanged and all configurations produce exactly the same final event trace. Furthermore, for a given number of partitions the committed events remain constant.

Such partitioning artifacts consistently impact all the models, particularly when a few partitions are used. However, as shown in Figure 5(a), the set of committed events rapidly stabilize with increase in the number of partitions as incremental change in number of overlapping agents rapidly diminishes. However, both `Split EcoArea` configurations continue to exhibit minor jitters on number of committed events with changes in number of partitions. In addition, `Config` #3 involving proxy agents consistently commits about 2000 events more than `Config` #2. These events are generated to implement activation and deactivation of proxy agents as described in Section 5. However, due to the scale of the graph this difference is not apparent in Figure 5(a).

Although `Config` #2 and `Config` #3 commit almost the same number of events, their performance is significantly different as shown in Figure 4. The difference arises due to the dramatic difference in the number of rollbacks as illustrated by Figure 5(b). Recollect from Section 5 that in WESE inter-factory interactions are the primary source of rollbacks. `Config` #3 suffers from fewer rollbacks because proxy migration maximizes local interactions and minimizes inter-factory interactions. This characteristic is evident from the number of inter-factory messages shown in Figure 5(c). Conversely, in `Config` #2 agents remain on the same factory and generate numerous inter-factory messages as they migrate. The inter-factory messages in-turn increase the number of rollbacks.

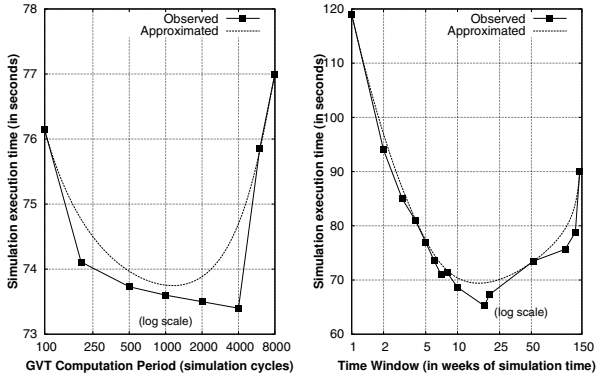
The charts in Figure 4 and Figure 5 clearly highlight the effectiveness of `Config` #3 involving `Split EcoAreas` and proxy agents. One of important aspects to note is that these experiments were conducted using optimal GVT computation intervals and an optimal time window for throttling the simulation. Throttling our simulations is necessary for two reasons. First, the agents do not stop life cycle activities and schedule events beyond the required end time. However, such events are unnecessary and the time window is necessary to minimize them. Secondly, the time window was necessary to throttle the simulations. Without the time window the number and length of rollbacks significantly increased causing dramatic degradation in performance. Similar observations have been reported by other researchers as well (Deelman and Szymanski 2002, Eubank 2002, Glass, Livingston, and Conery 1997, Maniatty, Szymanski, and Caraco 1999). Interestingly, the need for controlling optimism seems pervasive to spatially-explicit, optimistic simulations running on both shared and distributed memory architectures.

The impact of GVT period on simulation time is shown in Figure 7(a). Note that, smaller GVT periods are synonymous with more aggressive GVT computations and garbage collection. Figure 7(a) shows both the raw observations and the Bezier approximation of the data points. As expected,



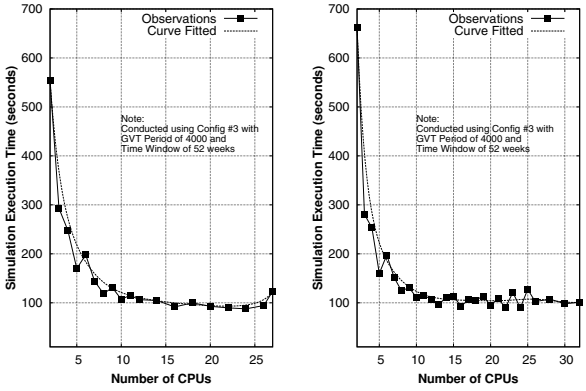
(a) Committed Events (b) Rollbacks (c) Inter-factory messages

Figure 5: Comparison of committed events, rollbacks, and inter-factory (over network) messages for model `M1`



(a) GVT Period (b) Time Window

Figure 7: Performance impacts of GVT period and throttling using a Time Window for model `M1`



(a) Model `M2` (b) Model `M3`

Figure 8: Simulation execution timings for larger models.

neither a too small nor a too large of a GVT period is optimal. The ideal GVT period was found to be between 2000 and 4000. This value was used for all experiments reported in this article.

The effect of changing the time window to throttle the optimism is shown in Figure 7(b). Smaller time windows result in more aggressive throttling curtailing the optimism. Conversely, larger time windows permit the simulation to optimistically advance; but rollbacks can be longer and more expensive. Similar to the GVT period, neither a small time window that curtails optimism nor a large time window that permits unrestrained optimism is advantageous. Figure 7(b) indicates that a broad range of window sizes are effective. We choose the median value from this graph as the optimal time window for all the simulations reported in this paper.

Having identified the optimal simulation configuration, we utilized them for simulating the larger models, namely `M2` and `M3` shown in Table 1. The simulation execution times for these two models using `Config` #3 is shown in Figure 8. Recollect that simulation execution times vary due to the nature of the model and the type of partitioning used. However, the approximated curve-fitted plot illustrates the general trend in the observations. As shown by the graphs in the figure, the simulations provide excellent scalability and performance. The empirical evidence highlights that the design goals of SEARUMS++ have been successfully met.

7 CONCLUSIONS

The application of simulation-based analysis is gaining momentum for epidemiological analysis of emergent diseases such as avian influenza. We have developed an Eco-modeling, parallel bio-simulation, and epidemiological analysis environment called SEARUMS++. It has been developed by completely redesigning and replacing the simulation infrastructure of an existing software system. The fundamental motivation for the redesign was to circumvent the scalability and performance bottlenecks of the earlier design. Scalability is important to meet the computational demands while performance is necessary to facilitate anal-

ysis of numerous scenarios in order to elicit comprehensive epidemiological knowledge.

This paper described in the issues involved in the design and development of a Time Warp synchronized parallel simulation back-end in C++. The different design alternatives that were incrementally developed and empirically evaluated to identify the optimal simulation configuration were discussed. The paper discussed the experiments, conducted on a distributed-memory super computing cluster to evaluate the effectiveness of the revised design. The best results were obtained from the design that used vertically split spaces and proxy agents to achieve dynamic, physical migration of agents. This configuration achieved best results by minimizing inter-factory events which are the primary source of rollbacks.

The experiments indicate that optimistic synchronization methodology effectively utilizes the latent parallelism in the model to yield improved scalability and performance. Furthermore, it is a strong evidence that the design objectives were successfully met. Nevertheless, the optimism had to be throttled to reach an optimal equilibrium between increased parallelism and rollback overheads. Literature review and our experience indicate that the need to throttle Time Warp simulations for spatially explicit models is pervasive to both shared memory and distributed memory architectures. Lastly, the success of our endeavor continues to emphasize the current and future importance of parallel simulations in epidemiology and related fields.

REFERENCES

Anderson, R. M., and R. M. May. 1992. *Infectious diseases of humans: Dynamics and control*. Walton Street, Oxford OX 2 6DP: Oxford University Press.

Brahmbhatt, M. 2006, June. Economic impacts of avian influenza propagation. In *First International Conference On Avian Influenza in Humans*.

CDC 2006, August. Centers for Disease Control & Prevention: Avian Influenza: Current Situation. Via <www.cdc.gov/avian/outbreaks/current.htm>.

Deelman, E., and B. K. Szymanski. 2002, March. Simulating spatially explicit problems on high performance architectures. *Journal of Parallel and Distributed Computing* 62 (3): 446–467.

Eubank, S. 2002, March. Scalable, efficient epidemiological simulation. In *Proceedings of the 2002 ACM symposium on Applied computing*, 139–145.

Ferguson, N. M., D. A. T. Cummings, C. Fraser1, J. C. Cajka, P. C. Cooley, and D. S. Burke. 2006. Strategies for mitigating an influenza pandemic. *Nature* 442:448–452.

Fujimoto, R. 1990, October. Parallel discrete event simulation. *Communications of the ACM* 33 (10): 30–53.

Glass, K., M. Livingston, and J. Conery. 1997. Distributed simulation of spatially explicit ecological models. In *PADS '97: Proceedings of the 11th workshop on Parallel and Distributed Simulation*, 60–63.

GLiPHA 2007. Global Livestock Production and Health Atlas (GLiPHA): Animal Production and Health Division of Food and Agriculture Organization of the United Nations. Via <www.fao.org/ag/ aga/glipha>.

GROMS 2006, April. Global Register of Migratory Species: Summarising Knowledge about Migratory Species for Conservation. Via <www.groms.de/>.

Hagemeijer, W., and T. Mundkur. 2006, May. Migratory flyways in europe, africa, and asia and the spread of HPAI H5N1. In *International Scientific Conference On Avian Influenza and Wild Birds*. Rome, Italy: FAO and OIE.

LANL 2006, April. Los alamos national laborartory: Avian flu modeled on supercomputer, explores vaccine and isolation options for thwarting a pandemic. Via <www.lanl.gov/news images/avianflu.shtml>.

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

Maniatty, W., B. K. Szymanski, and T. Caraco. 1999, January. High-performance computing tools for modeling evolution in epidemics. In *Proceedings of the 32nd Annual Hawaii International Conference on System Sciences*, 10–18.

Normile, D. 2006, March. Avian influenza: Evidence points to migratory birds in H5N1 spread. *Science* 311 (5765): 1225.

Rao, D. M. 2003. *Study of dynamic component substitution*. Ph. D. thesis, University of Cincinnati.

Rao, D. M., A. Chernyakhovsky, and V. Rao. 2007, October. SEARUMS: Studying epidemiology of avian influenza rapidly using simulation. In *Proceedings of ICMHA'07*, 667–673. Berkeley, CA, USA.

SEARUMS 2008, April. SEARUMS Website: WSC'08 Appendix. Via <www.searums.org/downloads/wsc08Appendix.pdf>.

SEDAC 2007, Dec. Socioeconomic Data and Applications Center (SEDAC): Gridded Population of the World. Via <sedac.ciesin.columbia.edu/>.

WHO 2006, July. World Heath Organization (WHO): Avian influenza. Via <www.who.int/csr disease/avian_influenza>.