Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 2187 – 2192

Procedia Computer Science

2013 International Conference on Computational Science

Parallel Programming Approaches for an Agent-based Simulation of Concurrent Pandemic and Seasonal Influenza Outbreaks

Milton Soto-Ferrari[a,]*, Peter Holvenstot[b], Diana Prieto[a], Elise de Doncker[b], John Kapenga[b]

[a]Department of Industrial Engineering, Western Michigan University, Kalamazoo MI 49008-5200, USA
[b]Department of Computer Science, Western Michigan University, Kalamazoo MI 49008-5200, USA

Abstract

In this paper we propose parallelized versions of an agent-based simulation for concurrent pandemic and seasonal influenza outbreaks. The objective of the implementations is to significantly reduce the replication time and allow faster evaluation of mitigation strategies during an ongoing emergency. The simulation was initially parallelized using the g++ OpenMP library. We simulated the outbreak in a population of 1,000,000 individuals to evaluate algorithm performance and results. In addition to the OpenMP parallelization, a proposed CUDA implementation is also presented.

Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science

Keywords: Pandemic Outbreak; Simulation; OpenMP; CUDA.

1. Introduction

It is currently impossible to predict the upcoming of a Pandemic Influenza (PI) virus, and its impact on the population and the public health systems. A recent example is the 2009 H1N1 PI, which differed significantly from the pandemic outbreak expected by public health authorities worldwide. It was believed that PI was going to begin in Asia, through the progression of the highly severe but mildly transmissible H5N1 bird flu. Instead,

* Corresponding author *E-mail address:* miltonrene.sotoferrari@wmich.edu

Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science
doi:10.1016/j.procs.2013.05.389

the PI initiated in Mexico, in the form of the H1N1 Swine Flu, which is highly transmissible but not as severe as the H5N1 virus. These H1N1 features were only recognized after seriously infected cases sought healthcare and were reported to the surveillance system [1].

Before the 2009 PI, several agent-based (AB) simulation models were created to understand the likely progression of the disease in the population, and the effect of mitigation strategies in the pandemic progression. Results from these models were used to create governmental guidelines for the mitigation and containment of the PI [1]. However, these models were created using the epidemiological features of the H5N1, and some recommendations were hard to follow for the real H1N1 PI. As an example, Pandemic guidelines recommended the closure of schools and workplaces once the outbreak was confirmed. But the H1N1 was too mild to close all the schools and workplaces of an area. In the U.S., a small percentage of locations were closed during the pandemic period. These lessons from the H1N1 PI, together with the views of federal, state and local decision makers [2] demonstrate that AB models are yet to be adapted to support operational decisions. Such decisions usually take place in cycles of 4 to 6 hours during the progression of a PI outbreak.

To support operational decisions, PI simulations must run extremely fast, even with the increase of the outbreak scale, to allow replication and retrieval of significant results within the operational decision cycle, i.e., 4-6 hours. These computational gains can be achieved through the use of parallel programming techniques. Several approaches have been proposed for parallelizing simulations of influenza outbreaks. DiCon [3], ABM++ [4], and FluTE [5] are examples of MPI implementations. The EpiSimdemics software is an example of a GPU approach that uses parallel blocks to support locations [6]. Those implementations are designed to work with AB simulations of one influenza virus, either pandemic or endemic.

In this paper, we present a parallelized version of the AB model described in [7], which simulates the disease spread of a pandemic and an endemic (seasonal) virus through an urban population. The main feature of the AB model is that it can easily accommodate the epidemiological features of seasonal and pandemic outbreaks while both outbreaks are occurring.

The parallelized version of the AB model was developed over the OpenMP parallel environment [8]. We evaluated the performance of the implementation for 1,000,000 inhabitants. In addition, we propose the features of a novel CUDA implementation, which is the subject of our future work.

The paper is organized as follows: In section 2, we present the basic and parallelized versions of the algorithm. The results are presented in section 3. The proposed CUDA implementation and the conclusions are presented in sections 4 and 5, respectively.

2. Agent-based model description

2.1. Basic version

The basic AB model simulates the spread of a Pandemic H1N1 flu virus, and one of its seasonal antigenic variants. Both viruses are seeded in an urban population that considers different locations, including schools, workplaces and errand places. The simulation was populated using information from the Hillsborough County in Tampa, Florida, USA. The data collected for the simulation were reported in [9], and were obtained from the 2002 U.S. Economic Census, the 2001 American Community Survey, and the 2001 National Household Travel Survey. For a population of 1,000,000, a total of 12,800 businesses, and 500,000 households were simulated.

The program can be generally described as shown in the flow diagram of figure 1. The diagram shows the following sequence: First, the program executes a setup routine where simulation parameters are read from text files. Then, a fixed number of businesses are generated and sized using the census information in the text files. A fixed number of households are then generated, with each household's size being assigned according to a census-based probability distribution. In addition, individuals are assigned their personal data including

workplaces and schools. Finally, the initial infection is spread to a fixed number of individuals, as symbolized by the box number 4 in figure 1.

Box number 5 shows the beginning of the main program loop. Each hour, the program iterates all locations including households, schools, and workplaces and moves all individuals across locations according to a daily schedule (boxes 6 and 6a). The locations are iterated again, and for each individual infected with either seasonal or pandemic viruses, the program generates a number of contacts with other individuals at the new location, (boxes 7 and 7a). At the end of the day, infected individuals are iterated, and each of their contacts, if susceptible, has a certain chance of contracting one or both viruses. Selected susceptible individuals then become infectious (boxes 8, 8a and 8b). In addition to the infection of individuals, already infected individuals are also evaluated, to determine if their culmination periods have elapsed and they can recover (boxes 8c and 8d). After the simulation has run to completion, individuals are iterated in order to ascertain the number of infected cases per primary case, i.e., the reproduction number [7]. The boxes representing an iteration of locations are printed in black.

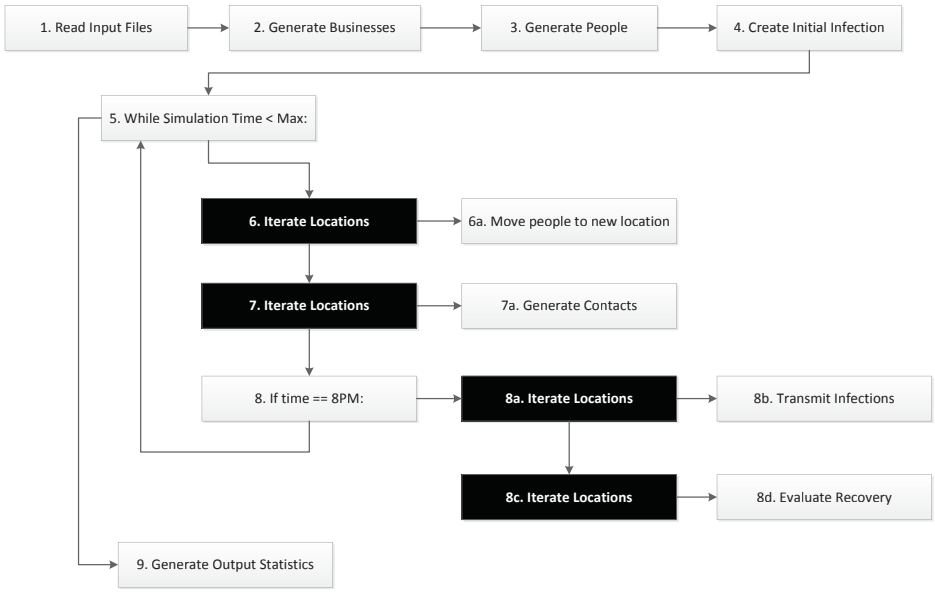

Fig. 1.Basic AB model

2.2. Open MP version

The main program loop was parallelized using an OpenMP parallel-for pragma clause (using a dynamic schedule with default chunk size for load balancing) before each of the for-loops used to iterate locations (boxes 6, 7, 8a and 8b of figure 1). Figure 2 shows the pseudocode of the program described in figure 1, and the additions to the basic code for enabling parallelization (in bold). Testing revealed the dynamic schedule to be the most efficient. Each location was equipped with an OpenMP mutual-exclusion lock in order to prevent race conditions. Using this method, a precise order of the individuals within a location array cannot be guaranteed; so replicates with the same RNG (random number generator) seed may have different outputs. This non-determinism should not affect the long run measures of central tendency and variability. As the computation time was inconsequential to the overall running time, the output calculations were not parallelized.

```
1: Read inputs ( )
2: Generate businesses ( )
3: Generate people ( )
4: Create initial infection ( )
5: While simulation time < max
6:       (Parallel) for each location
7:        Move people to new location
8:        (Parallel) for each location
9:        Individuals make contacts
10:       If hour = = 8pm
11:                (Parallel) for each location
12:                 Individuals transmit infection
13:                 (Parallel) for each location
14:                 Individuals become infectious or recover
15:                 Compute statistics and terminate
```

Fig. 2.OpenMP model

3. Results

Our test environment was the computer cluster in the Department of Computer Science Research laboratory, at Western Michigan University in Kalamazoo, Michigan. The cluster is a shared memory, multicore and GPU based system, with multi-core processor. The program was built in g++ -fopenmp compilation using Linux. Figure 3a shows the decrease in runtime from T_1 = 1060s when executed sequentially, to approximately T_{24} = 301s when the number of threads P equals the number of cores, with the OpenMP parallel version. This amounts to a speedup of $S_{24} = T_1/T_{24} = 3.52$ (figure 3b). Increasing the number of threads further (through 32) leaves the time fairly constant - actually reaching a minimum of 301s at 24 threads.

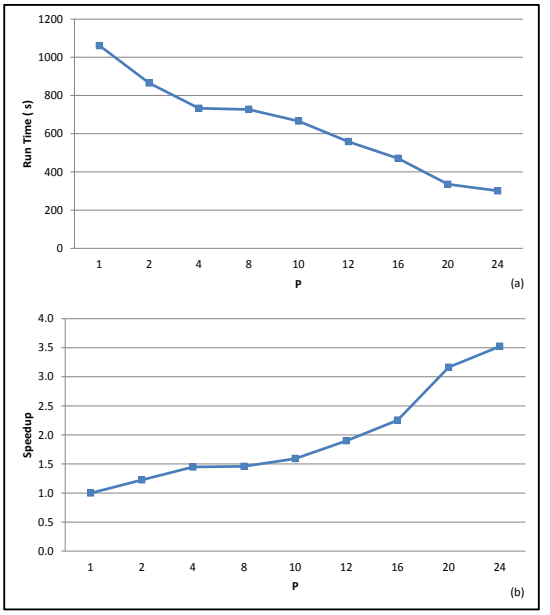

Fig. 3.OpenMP results

4. Proposed CUDA Approach

The most computationally intensive portion of the AB simulation is the modeling of interactions between the individuals within the locations. We initially tried to locate the household interactions within the GPU, while the host was running the workplace and errand interactions. But the running time for this implementation was slower than the times obtained with our OpenMP implementation. In addition, we faced load balancing issues with the scarce memory of the GPU.

We now propose a CUDA implementation of the model, which is expected to perform significantly faster than our OpenMP version of the code. As depicted in figure 4, the proposed CUDA approach generates the inputs and schedules for the agents on the host based on demographic data (we assume that data should have been collected for every population center before generating all the agents); hereafter the kernel is invoked to perform all interactions among the individuals on the GPU. Each block on the grid will support an AB simulation of a population center (e.g., a town, a county, a province or a more populated region). Synchronization among blocks will occur based on probabilistic assessments on the number of people traveling from one locality to the other. When the running time is complete, a reduce clause accumulates the value for the reproduction number of the disease. Then the value is sent back to the CPU for the analysis, calibration and evaluation of the results.

To develop the implementation, we will need to perform several experiments. First, we will need to determine the optimal "population center" size, which will be a function of the available information per population center (e.g., Census data) and the capabilities of the hardware. Second we will need to determine the optimal grid configuration to balance the load. Memory layout for coalesced memory access is also a design priority.

Our CUDA implementation will differ from existing approaches in the way the blocks within the GPUs are populated. Currently, there exist approaches where the blocks support locations within the GPUs [6]. The issue with this approach is that, if the blocks are designed to be large, there will be a large number of threads idling in blocking states for synchronization, which can be exacerbated by large differences in the number of individuals in each location. We believe that by working with population centers we will be able to reduce the number of idling threads and increase performance.

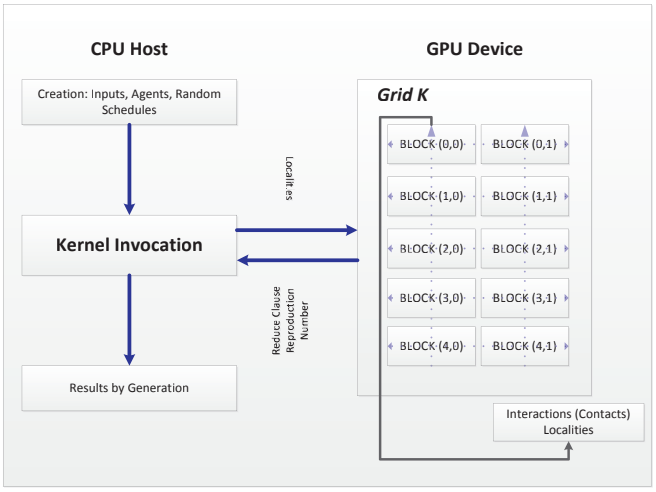

Fig. 4. Proposed CUDA implementation

5. Conclusions

In this paper, we described our parallel implementation of an AB model simulating Pandemic and Seasonal influenza outbreaks. Our preliminary results show significant reductions in the computational time for a population of 1,000,000 people. This is just a step in addressing the challenge of simulating outbreaks that are occurring in reality, and designing powerful decision support tools for operations during emergencies. For such a purpose, we designed and proposed a CUDA approach that could accommodate the exponential growth of a nascent outbreak.

This implementation can be extended to severe influenza outbreaks that are not necessarily pandemic, but could create stresses in the public health system. At the time of writing, the H2N3 outbreak of 2013 was rapidly escalating in the U.S., and thousands of patients were seeking healthcare and confirmatory testing. Our implementation could be useful in simulating such situation, if needed.

Acknowledgement. The authors acknowledge support by the National Science Foundation under grant number 1126438.

References

[1] Prieto D, Das T, Savachkin A, Uribe A, Izurieta R, Malavade S. A systematic review to identify areas of enhancements of pandemic simulation models for operational use at provincial and local levels. *BMC Public Health* 2012;**12**:251.

[2] Rosenfeld LA, Fox CE, Kerr D, Marziale E, Cullum A, Lota K, Stewart J, Thompson M. Use of computer modeling for emergency preparedness functions by local and state health official: a needs assessment. *J Public Health Manage Pract* 2009; **15**(2)**:**96-104.

[3] Dicon reference [http://www.bio.utexas.edu/research/meyers/dicon/].

[4] ABM++ reference [https://mission.midas.psc.edu/index.php?option=com_content&view=article&id=85&Itemid=115].

[5] Chao DL, Halloran ME, Obenchain VJ, Longini IM. FluTE, a publicly available stochastic influenza epidemic simulation model. PLoS Comput Biol 2010, **6**:1-8.

[6] Bisset KR, Aji AM, Marathe MV, Feng W. High-performance biocomputing for simulating the spread of contagion over large contact networks. *BMC Genomics* 2012; **13**(2):S3.

[7] Prieto D, Das T. An operational epidemiological model for calibrating agent-based simulations of co-circulating pandemic and seasonal influenza outbreaks. Unpublished paper.

[8] http://www.openmp.org, OpenMP website.

[9] Uribe A, Savachkin A, Santana A, Prieto D, Das, T. A predictive decision aid methodology for dynamic mitigation of influenza pandemics. Special issue on optimization in disaster relief. *OR Spectrum* 2011; 6 May: 1–36.