Parallel algorithm for simulating the spatial transmission of Influenza in EpiGraph

Gonzalo Martín
Universidad Carlos III
Madrid, Spain
gmcruz@arcos.inf.uc3m.es

Maria-Cristina Marinescu
Barcelona Supercomputing Center
Barcelona, Spain
maria.marinescu@bsc.es

David E.Singh
Universidad Carlos III
Madrid, Spain
desingh@arcos.inf.uc3m.es

Jesús Carretero
Universidad Carlos III
Madrid, Spain
jcarrete@arcos.inf.uc3m.es

ABSTRACT

This paper introduces an approach to modeling and simulating the propagation of flu-like infectious diseases over large, widely spread urban areas connected by transportation networks. We incorporate geographic location and a transportation model into our region-based, closed-world EpiGraph simulator to realistically model the movement of the virus between different geographic regions. The resulting simulator can assist in understanding how outbreaks propagate between far apart regions due to the movement of people outside their base location. This paper describes the MPI-based implementation of EpiGraph and its performance evaluation when simulating large-scale scenarios. We evaluate the simulator both on a distributed memory system and on a shared memory system.

Categories and Subject Descriptors

I.6.3 [**Computing Methodologies**]: Modeling and simulation—*Applications*; J.3 [**Life and Medical Science**]: Medical information systems.

General Terms

Algorithms, Performance.

Keywords

Simulation, computational epidemiology, parallel algorithms, MPI, distributed computing.

EuroMPI '13, September 15 - 18 2013, Madrid, Spain

1. INTRODUCTION

Today's interconnected world, in which movement of people between urban areas is an ordinary reality, creates an environment where infectious agents can propagate fast and far. Recent decades show the occurrence of global influenza pandemics originated in Asia (1957, A/H3N2 strain) and Latin America (2009, A/H1N1 strain) as examples of outbreaks related to the movement of people between continents [9, 10]. Understanding the dissemination patterns of viruses such as influenza, over large geographic regions, would help public health authorities to respond more efficiently to outbreaks.

One of the existing tools for modeling influenza propagation is EpiGraph [7], an epidemiological simulator that can predict the evolution of infections over short to medium time frames within closed urban regions. EpiGraph was validated by comparing the simulation results against the data from the New York State Department of Health Report (NYSDOH), with similar temporal distribution results for the number of infected individuals.

EpiGraph is implemented as a scalable, fully distributed application based on MPI, but it assumes that there are no new individuals introduced in the population. One important source of incoming population is those people who travel between different urban areas, and which—voluntarily or not—get in contact with the local population. Modeling this type of contacts is crucial to understand the effect that travel and commute have on the evolution of epidemics at a global level. The work we report on in this paper is an extension of EpiGraph that enables the efficient simulation of infection propagation within arbitrarily far apart urban areas interconnected via transportation networks. This extension involves the design and implementation of a new parallel algorithm based on MPI for EpiGraph.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 summarizes the main features of EpiGraph. Section 4 introduces our approach to simulating the spatial transmission of flu-like infectious diseases. Results and performance evaluation are presented in Section 5. Section 6 summarizes the paper with the conclusions.

2. RELATED WORK

Epidemiological simulators based on social contact network models have become the most extensively used approach to simulating the propagation of infectious diseases in heterogeneous human social networks. This is a realistic approach because the dynamics of the propagation of infectious diseases is tightly related to the structure and the characteristics of the network of connections between the individuals within a population [3]. Contact network-based simulators—such as EpiGraph—model the evolution of epidemics as an stochastic process.

EpiFast [5] is an MPI-based simulator which implements a SIR-like model for simulating the evolution of epidemics on heterogeneous social contact networks. However, these contact networks are generated at random and neither demographic or geographic information are used. The parallel implementation of EpiFast is based on the master-slave communication model, which increases the complexity of the communications and reduces the scalability of the algorithm when executing on many processors.

FluTE [6] is a individual-based simulation model of Influenza epidemics. The social contact network is built using a hierarchical structure of communities based on data extracted from the census. Social contacts within each community are not modeled realistically because are generated at random as uniformly mixing groups. FluTE is able to simulate large-scale scenarios and implements a transportation model based on information extracted from the air traffic routes in the U.S. However, the complexity of the MPI-based parallel algorithm increases when the size of the communities is very large, thus reducing the performance of FluTE when simulating large-scale scenarios.

EpiSimdemics [4] is an epidemiological simulator which integrates an efficient, MPI-based parallel algorithm for simulating the propagation of infectious diseases in realistic social networks. EpiSimdemics presents an individual-based approach in which social contact networks are generated using demographic data extracted from the census without considering a transportation model. EpiSimdemics is able to simulate very large populations of up to 100 million people, although it requires large computing resources. In contrast, EpiGraph requires lower computing resources to simulate large-scale scenarios with a more sophisticated epidemic model and a realistic social model which captures the transmission of an infection across different urban regions due to the movement of the population.

3. EPIGRAPH BACKGROUND

EpiGraph consists of two main components: the social model based on the contact network of the individuals in a population, and the epidemic model which captures the different stages of the infection. Each individual in the social model is represented by age, gender, race, and occupation. We represent the interactions as an *interconnection graph* in which each node represents an individual and each edge represents a time-dependent interaction between two individuals. Social interaction patterns are modeled using real information extracted from on-line social networks such as Enron and Facebook; we use real demographic information to represent the characteristics of the individuals. Daily interaction patterns are based on the occupation and the time and the day of the week. This reflects the fact that at different times individuals may interact with each other in different environments: at work, at home, during leisure time, or via spontaneous contacts.

The epidemic model is specific to the infectious agent under study, in our case, the Influenza virus. We extended the classic SEIR epidemic model to include additional states such as latent, asymptomatic, dead and hospitalized. One feature of EpiGraph is that it is possible to evaluate the effect of intervention strategies such as vaccination, school closing, and social distancing for non-worker individuals. Social distancing restricts the interaction of individuals by retaining them at home, which reflects closures of public facilities to mitigate the spreading of the disease.

We implemented EpiGraph based on the SPMD paradigm. Communication and synchronization operations are carried out using MPI, which enables an efficient execution both on shared memory, as well as on distributed memory architectures. SPMD-based applications require a workload partitioning strategy to distribute the data that will be used by the processes that execute in parallel. In order to perform large scale simulations, it is necessary to manage the interactions of millions of individuals. This makes EpiGraph not only computation-intensive, but also memory-intensive.

The interconnection graph is stored internally as a sparse matrix. To make better use of the memory, data structures are distributed among processes rather than replicated on each of them. The sparse matrix is partitioned by dividing the population in independent sets which are assigned to different processes. Other data structures such as vectors which store individuals' information—health status, age, or race—are partitioned using the same methodology. As a result, each process is only responsible for performing the simulation for the local individuals assigned to it.

3.1 The parallel algorithm

Algorithm 1 shows the pseudocode for the parallel algorithm implemented by the simulator. The iterative algorithm consists of four phases. The first phase (`L3`) consists in updating the status of every local individual v in the epidemic model. Following other approaches, we have modeled the effect of Influenza on humans by means of a state automaton [7]. The algorithm processes the status of each infected individual and evaluates the probability of transitioning to the next sickness stage. An infected individual stops transitioning when he has reached the immune, recovered, or dead state.

The second phase (`L5`) consists in computing the dissemination of the infectious agent from infected to susceptible individuals. For every connection of every local infected individual v the algorithm evaluates the probability that the infection will be transmitted through to its contacts ($contacts(v)$) in the social network. This probability depends on the type of connections between individuals, the time of the day, and the specific characteristics of the individual subject to being infected—such as gender or age.

If during the second phase a local individual gets infected, its state will be locally updated by process $rank$ in the next iteration of the algorithm. However, if this individual is not local to process $rank$, it is necessary to communicate his new state to the remote process r responsible for him, thus maintaining the integrity and consistency of the data in the simulation. The updated status of each newly infected, non-local individual is stored by each process in a data structure

Algorithm 1 EpiGraph parallel algorithm.

Input: $(rank, p, status, A_{rank})$ where $rank$ is the rank of the process in the default, global MPI communicator, p is the number of processes in the global communicator, A_{rank} is the partition of the social contact network, and $status$ is the health status of the individuals of the urban region.

Output: $(status)$ where $status$ is the updated status of individuals in the simulation.

```
1: for timestep = 1 → simulation_time do
2:   for each local individual v ∈ A_rank do
3:     UpdateStatus(status(v))
4:     if status(v) is infectious then
5:       ComputeSpread(v, contacts(v), new_infected)
6:     end if
7:   end for
8:   for each process r ∈ p where (r ≠ rank) do
9:     SendNewInfections(new_infected_{rank→r}, rank, r)
10:  end for
11:  if timestep % n then
12:    Interventions(status(A_rank), rank)
13:  end if
14: end for
```

($new\_infected$) which records the transmission of the disease to individuals who are local to a remote process.

Communication of newly infected, non-local individuals is performed during the third phase (`L9`) of the algorithm. Each process $rank$ uses MPI point-to-point primitives (`MPI_Send` and `MPI_Recv`) to communicate newly infected individuals who are local to each remote process r in the simulation ($new\_infected_{rank \to r}$). Communications are designed to overlap in time to minimize the communication overhead.

The fourth phase (`L12`) consists in evaluating both pharmaceutical and non-pharmaceutical interventions in order to mitigate the propagation of the infectious disease. Non-pharmaceutical interventions—such as closing schools or social distancing—are triggered when the number of infected individuals in the population surpasses a threshold. Collective MPI operations (`MPI_Allreduce`) are performed every n time steps to gather the number of local infected individuals from all processes and then distribute the result back to all processes.

4. SIMULATION OF WIDE AREAS

We have extended EpiGraph to cope with the propagation of Influenza-like infectious diseases over large geographic areas within which people are traveling for work and vacation purposes. To realistically model the transmission of the infection between urban regions, we enhanced the original EpiGraph social model to incorporate geographic location and a transportation model for the population between different urban regions. This approach allows us to study the spatial dynamics of the spread of Influenza at a global level.

As our case study we simulated the propagation of Influenza in and between the 24 most populated cities of Spain (Table 1). We used demographic information obtained from the National Institute of Statistics of Spain [2] to determine the distribution of the population in group types—workers, students, unemployed, households and elderly—, and individual information such as gender, age, and race. In the next section we explain our approach, then we describe how we implemented it in MPI.

4.1 Modeling interconnected urban regions

The transportation model reflects the movement of people between cities for work, study, or vacation, and is based on the gravity model proposed by Viboud *et al.* [10]. The gravity model calculates the volume of individuals who move between two locations depending on the population size of the origin and destination cities, as well as the distance between them. EpiGraph implements a transportation model in which the bi-directional flow is equal and calculated considering the destination city to be the larger one in the pair. Once the inter-city flows are calculated, we randomly select individuals from specific group types within the populations and move them for a specific period of time. We consider movement of workers and students for distances below the threshold specified by the gravity model for close regions (119 km); this reflects the daily commute to neighboring cities. We also consider the long-distance commute (above 120km) of workers that need to reside at a different location for several days in a row. Additionally, we consider people from any group type that move at any distance for several days for vacation purposes.

The geographical information that we integrate into EpiGraph includes latitude, longitude, and distance between urban regions, and was extracted from the Google Maps web service using the Google Distance Matrix API [1]. Although this work simulates urban regions that are spatially co-located within the same country, EpiGraph can be used to simulate very large-scale scenarios in which regions spawn different countries or continents.

4.2 Process mapping

The goal of the mapping task is to balance the workload of the application between the MPI processes. We balance the workload at two levels: the first one at the top level of the simulation (inter-region level), and the second one at the internal level of each urban region (intra-region level).

At the inter-region level, we balance the workload of the application taking into account the workload associated to each urban region, the available computing power of the platform, and the spatial locality of the data. We exploit the data locality of the application by mapping processes which execute on the same processor to the same urban region, thus minimizing the cost of the MPI communications. We balance the global workload by calculating the relative computing power (in $FLOPS$) of each processing element (PE). We consider each of the cores of modern multiprocessors as an independent PE. The relative computing power of each PE is computed by means of an offline microbenchmark. The load associated with each city is estimated taking into account its population and the number of contacts in its intercommunication graph. Then, we assign to each urban region a computing power which is proportional with its associated load. Figure 1 shows an example of applying this procedure for the configuration of a sample scenario consisting of the urban regions of Madrid, Barcelona, and Valencia when executing on 4 processors: 3 Quad core processors and 1 Dual core processor. First, we order the PEs by their physical location and calculate their relative computing power. Then we use a basic allocation strategy, allocating each PE to one process (P_n). Using this method, 8 processes are allocated for Madrid, 4 for Barcelona, and 2 for Valencia.

At the intra-region level, we balance the computation by redistributing the workload between the processes involved

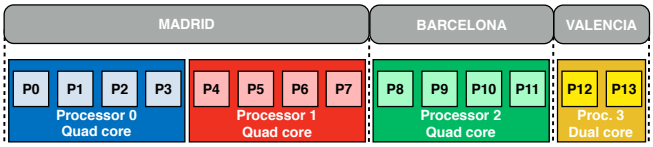


Figure 1: Process mapping according to the number of processing elements available for execution, the size of the urban populations, and exploiting the data locality.

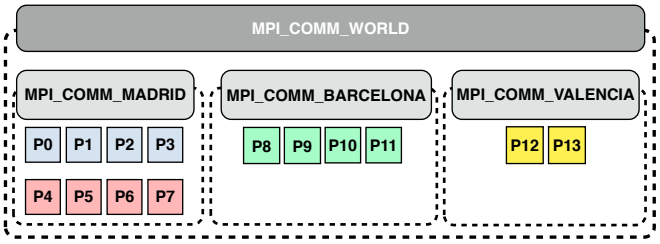


Figure 2: Two-level schema of MPI communicators.

in the computation of a specific urban region. We use the run-time library FLEX-MPI [8] to ensure that processes with more load transfer part of their assigned individuals (including their associated subgraphs in the interconnection graph) to the processes with less load within the same urban region. These operations are performed transparently, without user intervention.

4.3 MPI implementation

When executing an MPI application, all of the processes are by default grouped into the global communicator `MPI_COMM_WORLD`. A collective operation—such as those performed by EpiGraph when gathering the number of infected individuals within an urban region—that uses this global communicator blocks until all processes complete. In order to reduce the communication overhead of collective operations we used a communication model based on a two-level schema: the first level at the granularity of each urban region, and the second being the global default communicator.

The global communicator is used to perform communication operations between all of the running processes in behalf of the transportation model. Each process is identified by the communicator by a `global_rank`. Intra-region communications involve both point-to-point messages to transmit new infections and collective operations to gather the number of infected individuals. We introduce ad-hoc local communicators (`MPI_COMM_[REGION]`) to enable the decoupled execution between those subsets of processes that are associated with each urban region. Once the algorithm has mapped groups of processes to urban regions, the local communicators group together the subsets of processes involved in the computation of each specific urban region. Processes are identified in the `MPI_COMM_[REGION]` communicator by a `local_rank`. Figure 2 shows the two-level schema for the scenario described in Figure 1. This approach allows each region-specific subset of processes to perform collective operations independently of each other.

Algorithm 2 Spatial transmission algorithm.

Input: $(p, global\_rank, rp, local\_rank, urban\_regions, local\_region, A_{local\_rank}, status)$ where p is the number of processes in the global MPI communicator, $global\_rank$ is the rank of the process in the global communicator, rp is the number of processes in the local MPI communicator, $local\_rank$ is the rank of the process in the local communicator, $urban\_regions$ is the number of urban regions in the simulation, $local\_region$ is the urban region assigned to process $rank$ in the subset rp, A_{local_rank} is the partition of the social contact network of the urban region assigned, and $status$ is the health status of the individuals of the $local\_region$.

Output: $(status)$ where $status$ is the updated status of individuals in the simulation.

```
1:  for timestep = 1 → simulation_time do
2:    for each local individual v ∈ A_local_rank do
3:      UpdateStatus(status(v))
4:      if status(v) is infectious then
5:        ComputeSpread(v, contacts(v), new_infected)
6:      end if
7:    end for
8:    for each process r ∈ rp where (r ≠ local_rank) do
9:      SendNewInfections(new_infected_local_rank→r, local_rank, r)
10:   end for
11:   if timestep % n then
12:     Interventions(status(A_local_rank), local_rank)
13:   end if
14:   for each region ∈ urban_regions
      where (region ≠ local_region) do
15:     Transportation(pop_size(local_region), pop_size(region),
        distance, global_rank)
16:   end for
17: end for
```

4.4 Spatial transmission algorithm

Algorithm 2 shows the parallel algorithm that implements the spatial transmission of the infectious disease. In our implementation each urban region is simulated by a non-overlapping subset of processes rp within the set p of all processes. The social contact network of each urban region is partitioned between each subset of processes. Updating the status of local individuals (`L3`), the dissemination of the infectious disease (`L5`), the communication of newly infected individuals within the same urban region (`L9`), and evaluating interventions are computed in the same way as Algorithm 1. Note that intra-region communications are performed using the local communicator (`MPI_COMM_[REGION]`), where each process in the subset rp is identified by its `local_rank`.

The propagation of the infection via the transportation model (`L15`) is computed once a day for each pair of urban regions in the simulation. Each subset of processes corresponding to a region compute the number of individuals which move from this region to another region depending on the size of the two populations ($pop\_size(local\_region)$, $pop\_size(region)$) and the geographical distance between the locations ($distance$). Note that inter-region communications are performed using the global communicator (`MPI_COMM_WORLD`), where each process in the number p of processes is identified by its `global_rank`.

5. RESULTS AND DISCUSSION

We evaluated EpiGraph by simulating the spatial transmission of the flu virus both on a distributed memory system and on a shared memory system. The distributed platform is a cluster with 16 compute nodes, each of them has one

Table 1: Largest urban regions of Spain and number of processes (P) assigned to each urban region in the simulation.

City	Population	P	City	Population	P
Madrid	3,233,527	16	Valladolid	311,501	2
Barcelona	1,620,943	8	Vigo	297,355	1
Valencia	797,028	4	Gijón	277,733	1
Seville	702,355	4	Hospitalet	257,057	1
Zaragoza	679,624	4	A Coruña	246,146	1
Málaga	567,433	3	Vitoria	242,223	1
Murcia	441,354	2	Granada	239,017	1
Palma	407,648	2	Elche	230,587	1
Las Palmas	382,296	2	Oviedo	225,973	1
Bilbao	351,629	2	Badalona	220,977	1
Alicante	334,678	2	Cartagena	216,655	1
Córdoba	328,841	2	Terrasa	215,678	1

Intel Quad Core Xeon E5405 processor running at 2.00GHz and 4GB of memory. The shared memory system consists of a single compute node which has four Intel Xeon E7-4807 processors with Hyper-Threading support and 6 cores each, running at 1.87GHz and 128GB of memory. All the compute nodes run under Linux Ubuntu Server 10.10 with 2.6.35-32 kernel and are interconnected by a Gigabit Ethernet network. We use the MPICH-2 v1.4.1 implementation of MPI.

5.1 Large-scale area simulations

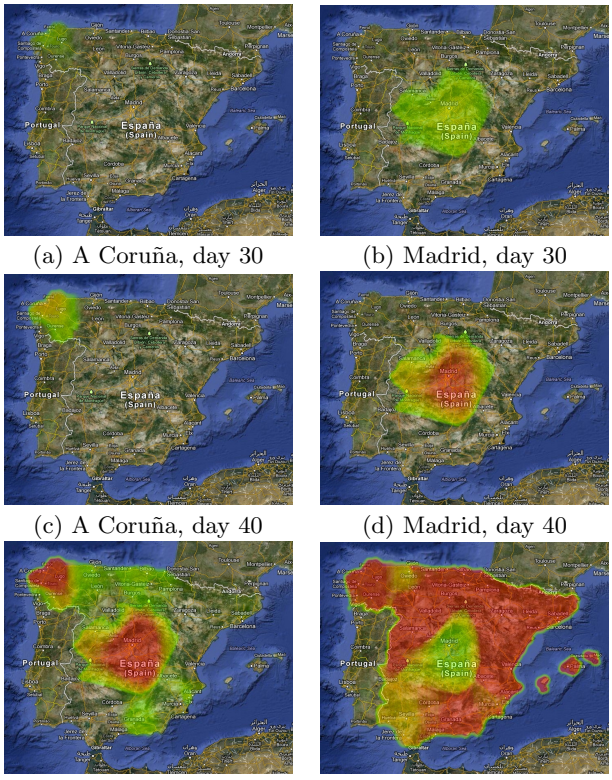
We simulated the virus propagation for the 24 most populated cities in Spain 1 and a simulated time span of 200 days. We executed the scenario on the cluster using 64 processes—4 processes per compute node. We compare the spatio-temporal propagation of the infectious disease when the outbreak is originated in different regions.

Figure 3 illustrates the spatio-temporal propagation of Influenza epidemics started in Madrid and A Coruña. We observe that the infectious disease is rapidly propagated when the epidemic is originated in a highly populated, well connected region (Madrid) compared with a smaller, more isolated region (A Coruña). When the epidemic is originated in Madrid the disease is rapidly propagated both to neighboring regions and to far apart regions due to the travel volume but also to the more pronounced long distance travel. Otherwise, when the epidemic is originated in an isolated region as A Coruña the virus takes several weeks to reach far away regions.

5.2 Performance evaluation

The following experiment evaluates the performance of EpiGraph when executing on both distributed and shared parallel architectures. We simulated the spreading of the virus in a medium-scale scenario which consists of a subset of 4 urban regions of the former 24: Madrid, Barcelona, Valencia, and Seville. This configuration allows us to evaluate the *strongscaling* of EpiGraph by increasing linearly the number of processes for executing the fixed-size, medium-scale scenario. Table 2 shows the number of processes mapped to each region when executing with 8, 16, 32, and 64 processes.

The execution of the medium-scale simulations requires a minimum of 2 compute nodes (running 4 processes each) when executing on the cluster due to the large memory footprint of the simulator. Thus, we consider the execution with 8 processes as our base execution in both systems.



(a) A Coruña, day 30 (b) Madrid, day 30

(c) A Coruña, day 40 (d) Madrid, day 40

(e) A Coruña, day 70 (f) Madrid, day 70

Figure 3: Comparison of the spatio-temporal propagation of Influenza at days 30, 40, and 70 since the outbreak of epidemics originated in A Coruña and Madrid.

Table 2: Process mapping for EpiGraph simulations, where NP stands for the number of processes and (P_M, P_B, P_V, P_S) stand for the number of processes assigned to Madrid, Barcelona, Valencia, and Seville, respectively. $Nodes$ stands for the number of compute nodes used when executing on the distributed memory system.

NP	P_M	P_B	P_V	P_S	Nodes
8	4	2	1	1	2
16	8	4	2	2	4
32	16	8	4	4	8

The experiment is bounded from above by 32 processes in the shared memory system because it has 48 logical PEs supported by Hyper-Threading and we cannot map 64 processes. Figure 4 shows how the application scales on both parallel architectures with respect to the performance of the base execution. EpiGraph scales well up to 32 processes in the cluster and almost linearly in the shared memory system due to a low intra-node communication overhead and a better cache behavior. The memory access pattern in EpiGraph is irregular because the processes access non-consecutive entries of the sparse matrix. When the number of processes increases there are less data assigned to each PE, which leads to a better cache behavior.

To analyse the performance in greater detail, we profiled

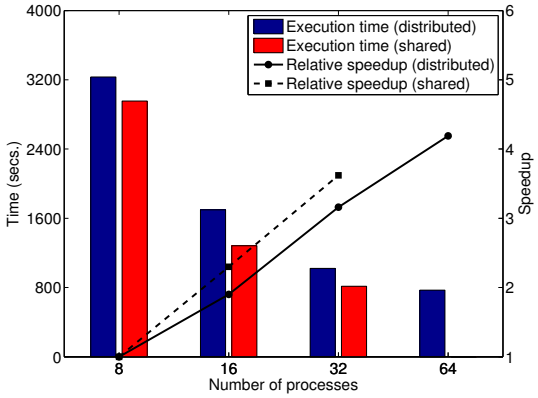


Figure 4: Execution time (left Y axis) and relative speedup (right Y axis) of EpiGraph.

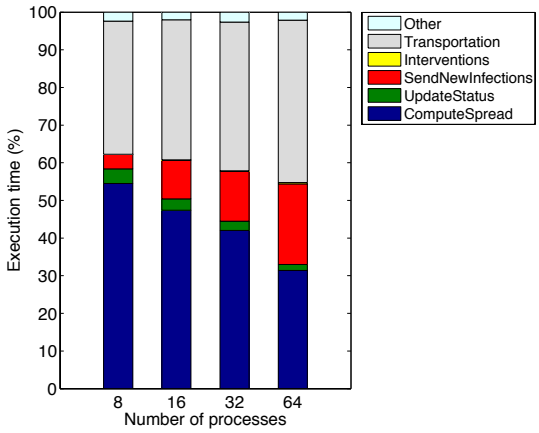


Figure 5: Time spent by EpiGraph in each of the phases of the simulation algorithm.

EpiGraph by instrumenting the code with wall-clock timing functions to collect the time spent by each process when executing on the cluster in each of the phases of the algorithm: (1) computing the dissemination of the virus (line `L5` in Algorithm 2), (2) updating the status of the local individuals (`L3`), (3) communicating newly infected individuals (`L9`), (4) evaluating interventions (`L12`), and (4) simulating the transportation model (`L15`). Figure 5 shows the percentage of the execution time which corresponds to the average time spent by all the processes in each of these phases. As we expected, the percentage of the execution time spent in the computation phases (*ComputeSpread* and *UpdateStatus*) decreases and the time spent in the communication phases (*SendNewInfections*, *Interventions* and *Transportation*) increases when using more processes for simulation. When executing on 32 and 64 processes more than half of the execution time is invested in communication operations. Note that in the simulations the execution time of the *Transportation* phase is significantly larger than the execution time of the *SendNewInfections* and *Interventions* phases. The simulation of the transportation model involves both collective and point-to-point communication and synchronization operations between all of the running processes, which increases the cost of the inter-region communications. Communication operations in the *Interventions* phase are performed using the ad hoc local communicators, which optimizes the cost of the collective intra-region communications consuming less than 1% of the execution time.

6. CONCLUSIONS

This paper presents a novel approach to simulating the spatial transmission of Influenza over large-scale areas. We have extended EpiGraph by implementing a new MPI-based parallel algorithm for simulating a transportation model, which allows us to study the spatial dynamics of the spread of Influenza. We use a two-level schema of MPI communicators to optimize the communications between processes. We have devised a process mapping strategy which considers two levels of parallelism to exploit the locality of the data and balance the workload of the application. Results show the high performance of EpiGraph when simulating large-scale scenarios both on a cluster platform and on a shared memory compute node.

7. ACKNOWLEDGMENTS

This work has been partially supported by the Spanish Ministry of Science under the grant IPT-430000-2010-14.

8. REFERENCES

[1] *Google Maps API. developers.google.com/maps/.*
[2] *National Statistics Institute (INE). www.ine.es/.*
[3] R. Anderson, R. May, and B. Anderson. *Infectious diseases of humans: dynamics and control*, volume 28. Wiley Online Library, 1992.
[4] C. L. Barrett, K. R. Bisset, S. G. Eubank, X. Feng, and M. V. Marathe. EpiSimdemics: an efficient algorithm for simulating the spread of infectious disease over large realistic social networks. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, page 37. IEEE Press, 2008.
[5] K. Bisset, J. Chen, X. Feng, V. Kumar, and M. Marathe. Epifast: a fast algorithm for large scale realistic epidemic simulations on distributed memory systems. In *Proceedings of the 23rd international conference on Supercomputing*. ACM, 2009.
[6] D. Chao, M. Halloran, V. Obenchain, and I. Longini. FluTE, a publicly available stochastic influenza epidemic simulation model. *PLoS computational biology*, 6(1):e1000656, 2010.
[7] G. Martín, M. Marinescu, D. Singh, and J. Carretero. Leveraging social networks for understanding the evolution of epidemics. *BMC Syst Biol*, 5(S3), 2011.
[8] G. Martín, M. Marinescu, D. Singh, and J. Carretero. FLEX-MPI - Technical Report. Technical report, Universidad Carlos III de Madrid, 2012. *www.arcos.inf.uc3m.es/∼desingh/publications.html*.
[9] S. B. Thacker. Spatial aspects of influenza epidemics. *The Journal of the American Medical Association*, 258(18):2593–2594, 1987.
[10] C. Viboud, O. N. Bjørnstad, D. L. Smith, L. Simonsen, M. A. Miller, and B. T. Grenfell. Synchrony, waves, and spatial hierarchies in the spread of influenza. *Science*, 312(5772):447–451, 2006.