Loimos: A Large-Scale Epidemic Simulation Framework for Realistic Social Contact Networks

Joy Kitson[†], Ian Costello[†], Diego Jiménez[*], Jiangzhuo Chen[‡], Jaemin Choi[¶], Stefan Hoops[‡], Esteban Meneses[*], Tamar Kellner[†], Henning Mortveit[‡], Jae-Seung Yeom[‖], Laxmikant V. Kale[¶], Madhav V. Marathe[‡,§], Abhinav Bhatele[†]

[†]Department of Computer Science, University of Maryland, College Park, MD, USA
[*]National High Technology Center, Costa Rica
[‡]Biocomplexity Institute and Initiative, University of Virginia, Charolettesville, VA, USA
[§]Department of Computer Science, University of Virginia, Charolettesville, VA, USA
[¶]Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
[‖]Lawrence Livermore National Laboratory, Livermore, CA, USA

I. Introduction

The past two years have shown how global pandemics can wreak havoc and inflict significant social, economic and personal losses. However, properly tailoring public health responses that could help mitigate those losses to circumstances on the ground is far from easy, and requires studying the potential impact and efficacy of a wide range of measures in a chaotic human-disease system. Simulating epidemic diffusion and possible interventions can help us in this goal. Agent-based models provide one method which has been used effectively in the past to model contagion processes, like diseases. We present Loimos, a highly parallel simulation of epidemic diffusion written on top of the Charm++ asynchronous task-based system. Loimos uses a hybrid between time-stepping and discrete-event simulation to model disease spread. We demonstrate that our implementation of Loimos is able to scale on to large core counts on different HPC platforms such as Theta at ALCF and Cori at NERSC.

II. Methodology

A. Model

We use an alternative approach to the coupled-rate equations commonly used in traditional compartmental models. Our method uses a combination of network theory, discrete event simulations, and agent-based modeling to study epidemics.

In Loimos, we model population behavior in terms of scheduled visits by people to locations. In order to determine which people are at the same location at the same time, we run a separate discrete event simulation at each location and maintain lists of infectious and susceptible people throughout the day. When a person leaves a location, they have a chance to infect each susceptible person currently there if they are infectious (or be infected by each infectious person currently there, if they are susceptible). Once a person is infected, they progress through various disease states stochastically in the manner prescribed by a disease model representing the disease being modeled.

B. Parallel Implementation

While each step of the discrete event simulation technically depends on every prior step at every other location, we can treat visits and infections taking place on the same day as independent of each other if we assume that the incubation time for the disease being modelled is at least a day (which is generally true for any real disease). This allows us to parallelize across people and locations within a given day.

In order to implement this parallel scheme, we used Charm++, a task-based parallel Framework built around combined work-data units called chares [1]. We use two distinct sets of chares in our implementation, one for people and one for locations, and perform computations in three phases, each of which is parallelized across the relevant chares. First, the people chares read in each person's visit schedule for the day and send visit messages. Next, the location chares process all visits to each location for the day and send interaction messages describing their chances of being infected. Lastly, the people chares process all interactions for each person, determine whether or not they were infected, and update their disease state.

TABLE I
Populations used in synthetic scaling experiment (SYN) and real data used in the intervention case studies.

Dataset name	visits	people	locations
REAL (CoC)	1,332,029	41,119	19,203
SYN (MD)	32,500,000	6,250,000	1,254,400
SYN (CA)	202,800,130	39,000,025	7,225,344
SYN (US)	1,715,829,570	329,967,225	80,281,600

C. Experimental Setup

We test the scalability on two separate HPC platforms: Cori [2] at Lawrence Berkeley National Lab and Theta [3] at Argonne National Lab. Both platforms are Intel-based Cray XC40 systems with an Aries dragonfly interconnect [4], but have different node configurations. Notably Cori consists of

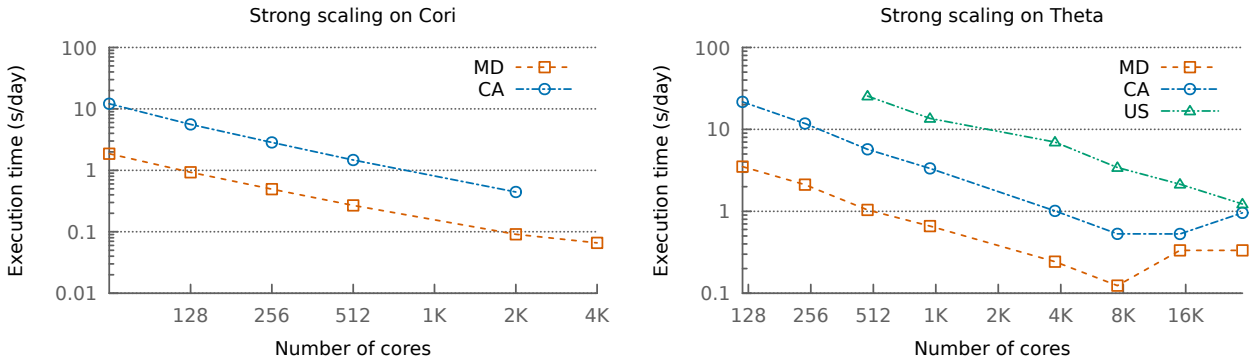

Fig. 1. Strong scaling performance of Loimos on Cori (left) and Theta (right) for three different datasets of 6.25, 39, and 330 million people.

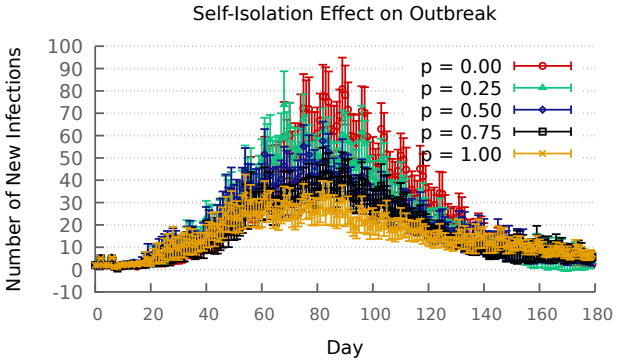

Fig. 2. Case Study Of Self-Isolation Intervention In Real CoC Dataset

two 16-core Haswell processors per node while Theta has a single 64-core Xeon Phi processor. All experiments were ran with Protobuf version 3.14.0 and Charm version 6.10.2.

We begin by performing extensive scalability studies using the best Charm parameters we found on both systems. Using a synthetic graph generation scheme, we create 3 realistically-sized datasets: two based on the states of California and Maryland, and one for the entire United States. Table I describes the number of people, locations, and total visits for all these datasets and for the real dataset used for the case study. We run the California and Maryland datasets for 180 days total and the US dataset for 30 days.

Lastly, we perform an intervention case study using a real dataset collected on the city of Charlottesville (CoC) in Virginia, USA. This dataset is based on surveys, phone data, and other anonymous sources and represents a week of realistic social interactions. We simulate case infectious rates depending on differing levels of compliance to a self-isolation policy. Individuals self-isolate (cease all interactions) if they have been infected and show symptoms. Note that individuals can be infectious yet asymptomatic; these people will continue to spread the disease even with 100% compliance. We test compliance levels from 0% to 100% with this intervention.

III. Results

In order to understand how Loimos would enable large scale simulations we performed classical scaling analysis. Figure 1 shows the strong scaling results of the simulator when running the three synthetic data sets listed in Table I on Cori and Theta. The general trend for the MD and CA datasets on both systems shows that the speedup keeps increasing up to 128 nodes. On Theta, for both datasets, problem size over-decomposition and communications costs seem to impact performance after 256 nodes. On Cori, an approximate speedup of 28 is obtained for both MD and CA datasets when running on $4k$ cores. On Theta a 40.81 acceleration is achieved for the CA dataset when running on $8k$ cores (128 nodes).

As for the US-sized synthetic data set, we started scaling from 8 nodes due to memory constraints on lower node counts. The trend observed on Theta is positive as speedup keeps increasing significantly up to $30k$ cores (512 nodes) achieving a 20.76 speedup. Following the acceleration curve, bigger speedups are expected for higher node counts for this dataset.

Figure 2 demonstrates the daily number of infectious individuals under a self-isolation intervention given five different levels of compliance. This intervention depends on each person's characteristics and demonstrates how a intervention could change people's schedules based on their disease status.

IV. Conclusions

In this work, we outlined the methods we used to develop this simulation framework and to optimize it for production HPC systems. We described the models underpinning our work as well as various optimizations we have made to enable the code to scale well. We demonstrate our code's efficient use of resources during strong scaling runs on the NERSC Cori and ALCF Theta supercomputers, and show how the application can be used to model the impact of interventions on the progression of an outbreak. Together, these runs demonstrate the potential uses of Loimos for policy makers as a fast, scalable epidemic simulator which is robust enough to capture the effects of policy interventions.

ACKNOWLEDGMENTS

This material is based in part upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Department of Energy Computational Science Graduate Fellowship under Award No. DE-SC0021. It is also based in part upon work supported by the National Science Foundation under Grant No. CCF-1918656. This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357, and those of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231 using NERSC award DDR-ERCAP0018674.

REFERENCES

[1] L. Kalé and S. Krishnan, "CHARM++: A Portable Concurrent Object Oriented System Based on C++," in *Proceedings of OOPSLA'93*, A. Paepcke, Ed. ACM Press, September 1993, pp. 91–108.

[2] "Cori system." [Online]. Available: https://docs.nersc.gov/systems/cori/

[3] "Theta system." [Online]. Available: https://www.alcf.anl.gov/alcf-resources/theta

[4] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *2008 International Symposium on Computer Architecture*. IEEE Computer Society, 2008.