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

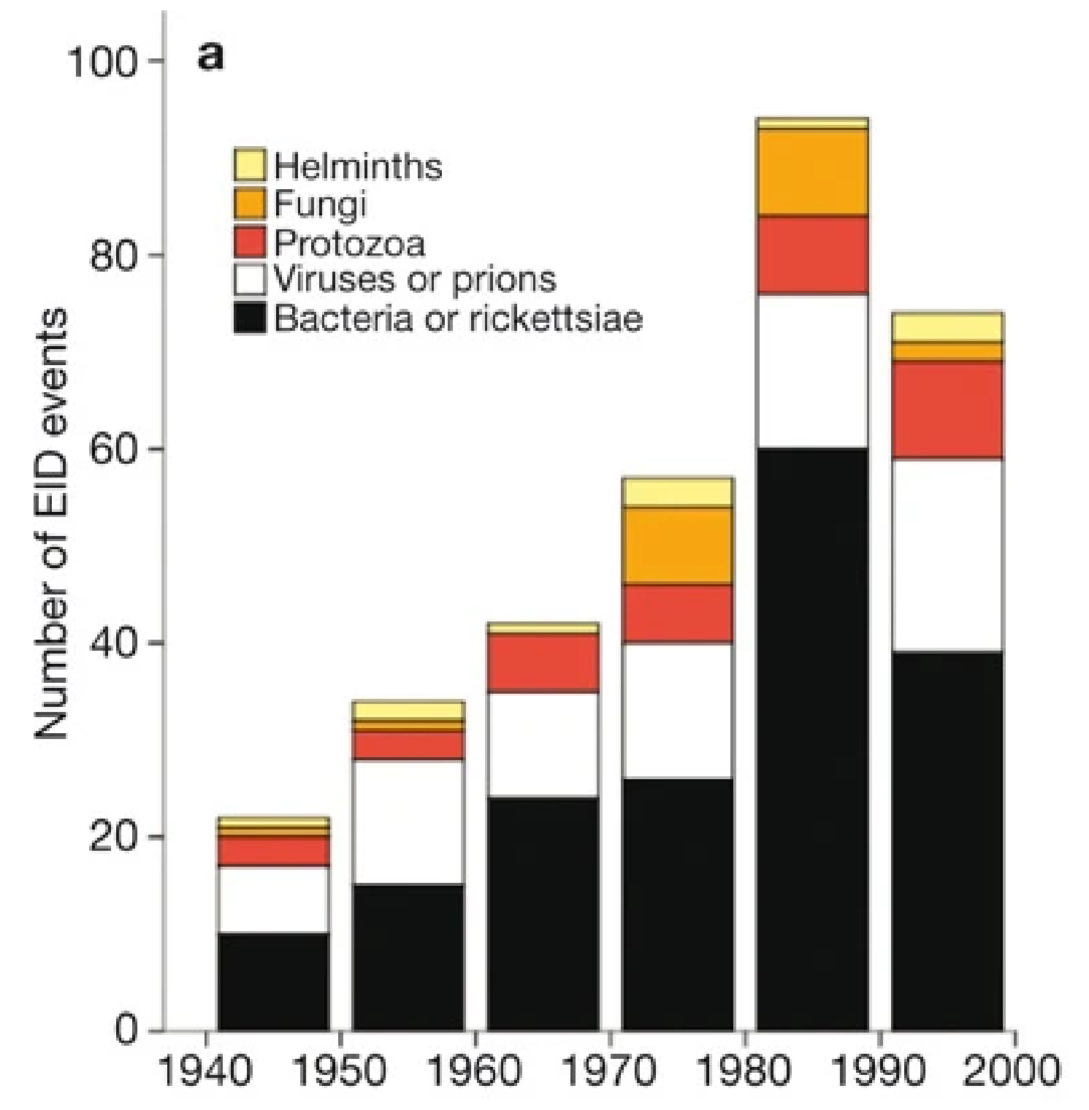
Joy Kitson[1], Ian Costello[1], Diego Jiménez[2], Jiangzhuo Chen[3], Jaemin Choi[4], Stefan Hoops[3], Esteban Meneses[2], Tamar Kellner[1], Henning Mortveit[3], Jae-Seung Yeom[5], Laxmikant V. Kale[4], Madhav V. Marathe[3], Abhinav Bhatele[1]
[1]University of Maryland, [2]National High Technology Center, [3]University of Virginia, [4]University of Illinois, [5]Lawrence Livermore National Laboratory

Motivation

- COVID-19 has made the costs of the spread of infectious diseases all too clear
- We need to be ready for the next outbreak, whether a new COVID-19 variant or an emerging infectious disease (EID)
- Responding quickly and intelligently will require modeling a variety of intervention scenarios in a short period of time
- We set out to design a scalable simulation of epidemic diffusion to meet that need

In order to inform policy decision effectively, an infectious disease model needs to:

1. simulate large populations
2. handle flexible interventions
3. account for uncertainty with large numbers of replicates
4. do all the above while maintaining a quick turn around time (at most a day)



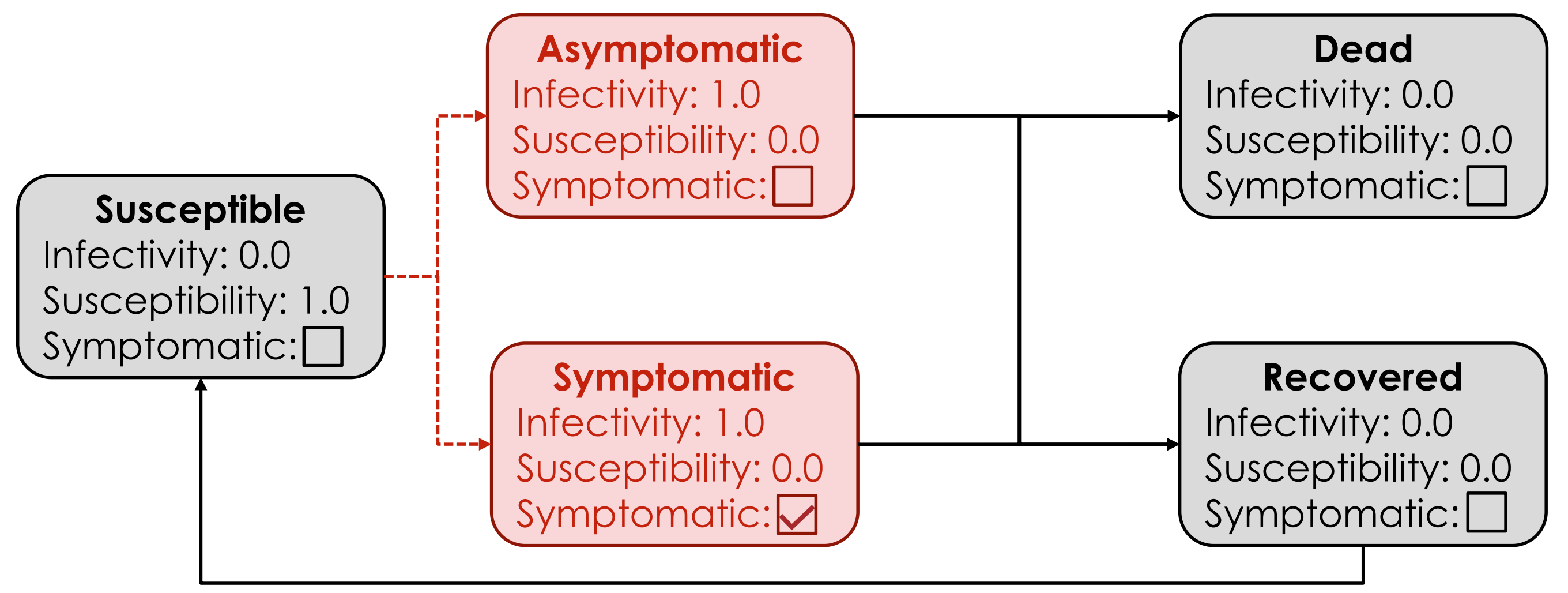
The rate at which new emerging infectious diseases (EIDs) appear is increasing. Taken from [1].

Model

We represent diseases using finite state automata (FSA):

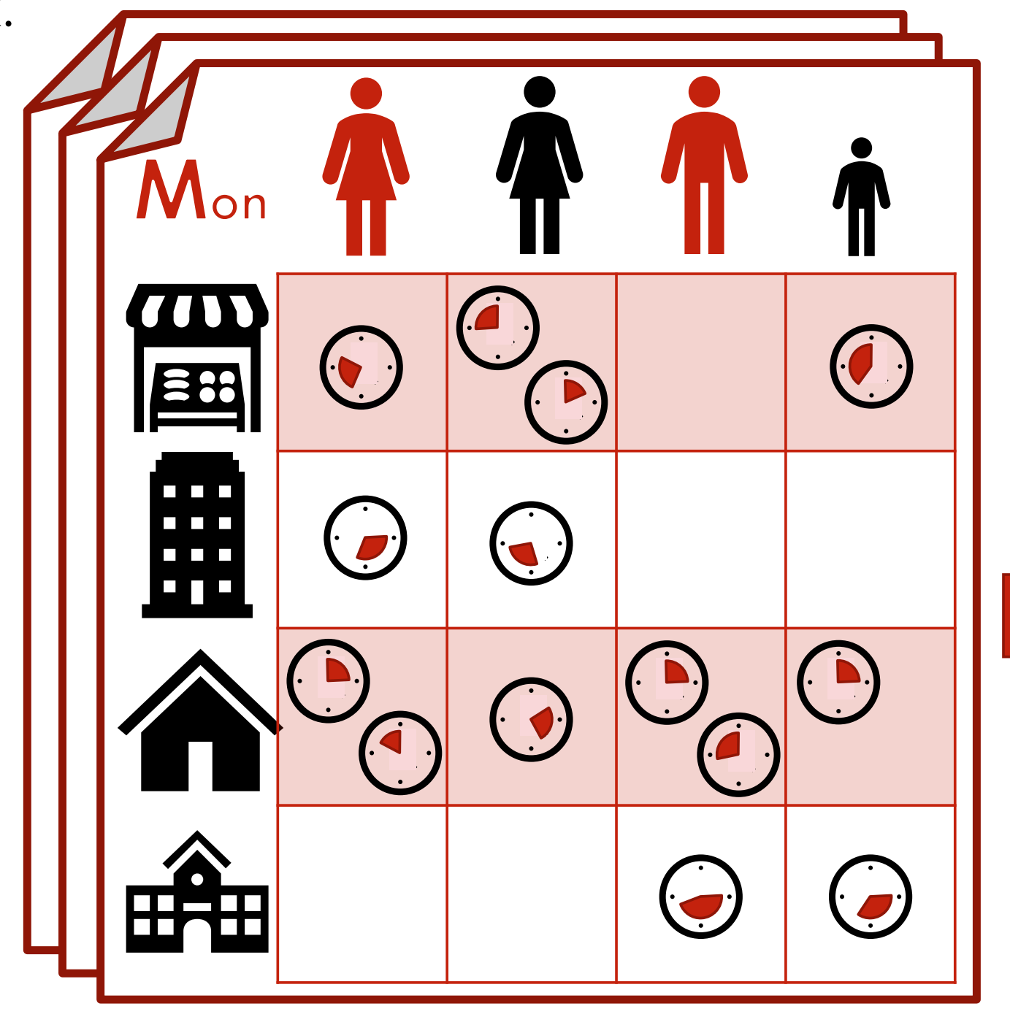
- Each state represents a different stage in the progression of the disease
- Each person maintains a disease state throughout the course of the simulation
- This state determines whether or they can infect – or be infected by – other people

- Every person starts in a *susceptible state*, moves to an *exposed state* after being infected, and progresses through subsequent states stochastically
- Simulating a new disease is as simple as making a new FSA to represent it

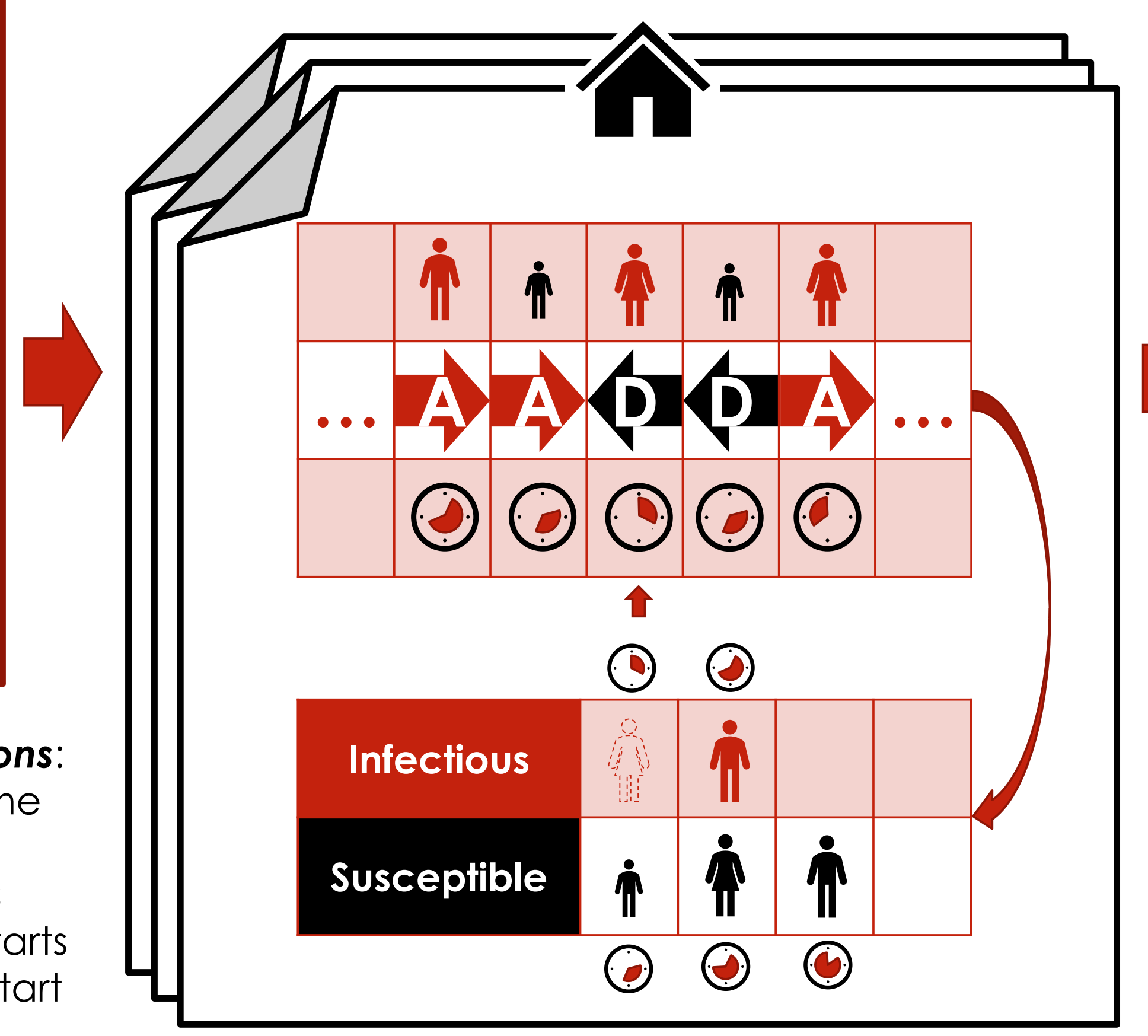


Model (continued)

1. Each person has a ***schedule*** detailing which locations they visit and when on each day of the week.

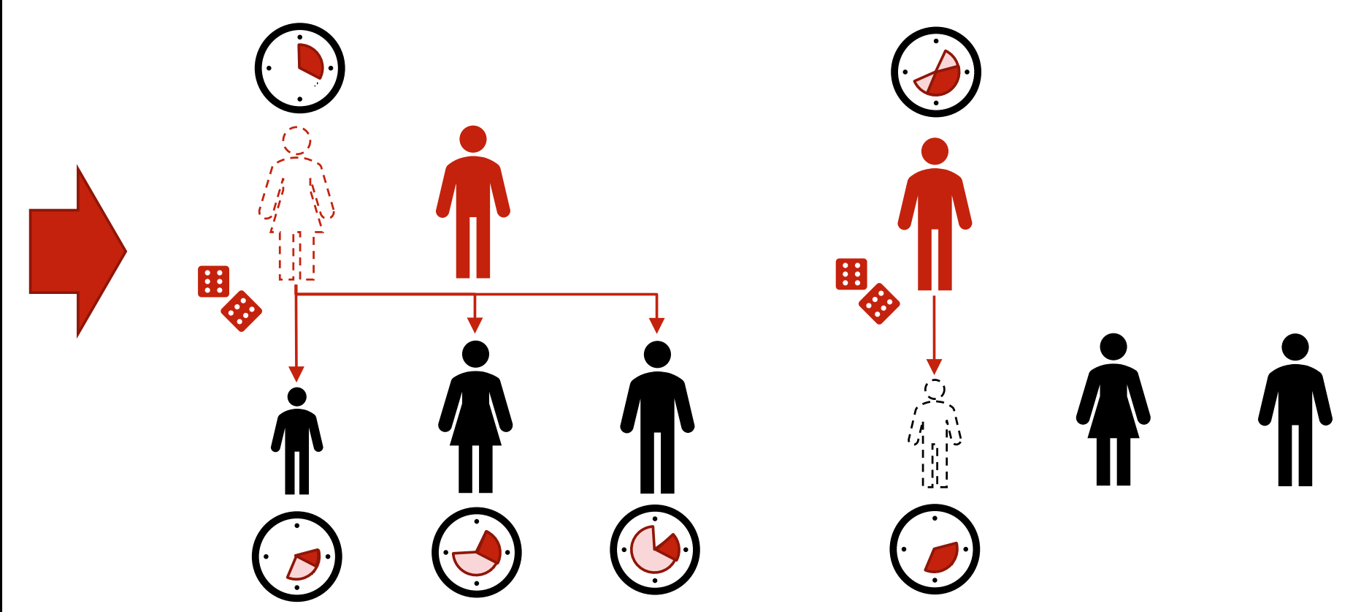


2. On each simulated day, locations receive a list of visits, then process the **arrivals** and **departures** in order of occurrence, keeping track of who is at the same location at the same time.



3. When each person leaves a location, we calculate the likelihood that they infected someone else or were infected themselves:

- **Infectious people** have a chance of infecting each susceptible person at the location when they leave
- **Susceptible people** have a change of being infected by each infectious person at the location when they leave



The simulation model relies on several ***assumptions***:

- People will not become infectious on the same day they are infected
- All transitions between disease states besides initial infections are known when each day starts
- Each person's visit schedule is known at the start of each day

Interventions can act on these models in several ways:

- When a person meets some criteria, they adjust their visit schedule
- When a location meets some criteria, visits to that location are adjusted
- When a person meets some condition, their disease state is changed to one with a different infectivity or susceptibility

Parallel Implementation

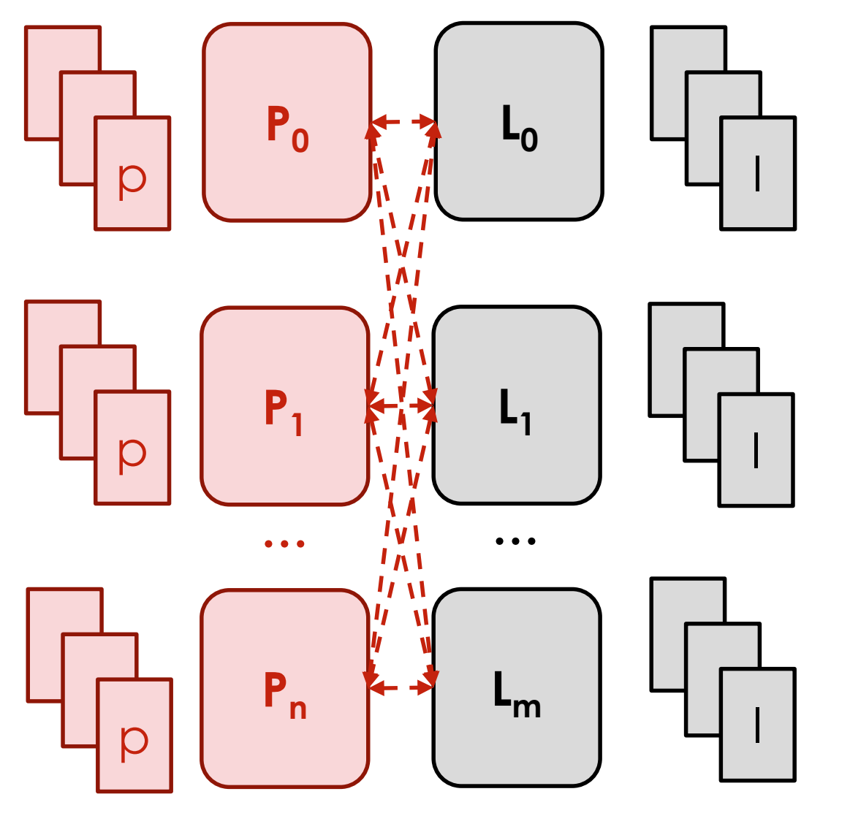
We implement Loimos in the Charm++ parallel framework. Charm++ is built around organizing code into combined work-data units called ***chares***. In Loimos, we use two types of chares:

People chares:

- Send visit messages
- Process interactions to determine if an infection occurs
- Update disease states

Location chares:

- Process visits
- Compute infection likelihood
- Send interaction messages



Loimos Algorithm

```
for each day:
  for each people chare pc:
    for each person p:
    | pc.send(p.visits)
  ------------------------------
  for each location chare lc:
    for each location l:
    | visits = lc.receive(l)
    | intrs =
    |  find_interactions(visits)
    | lc.send(intrs)
  ------------------------------
  for each people chare pc:
    for each person p:
    | if p.is_infected():
    | | p.update_state()
    | else:
    | | intrs = pc.receive(p)
    | | was_infected =
    | |   process_interactions(intrs)
    | | p.update_state(was_infected)
```

Experimental Design

We perform three experiments:

- Two ***strong scaling*** studies, on Cori at NERSC and Theta at ALCF
- An ***intervention*** case study on self-isolation, with varying levels of compliance

Name	Architecture	CPU	Cores/Node	Mem/Node	Network
Cori	Cray XC40	Intel Xeon E5-2698 v3	32	128 GB	Aries
Theta	Cray XC40	Intel Xeon Phi 7230	64	192 GB	Aries

Systems used for experiments

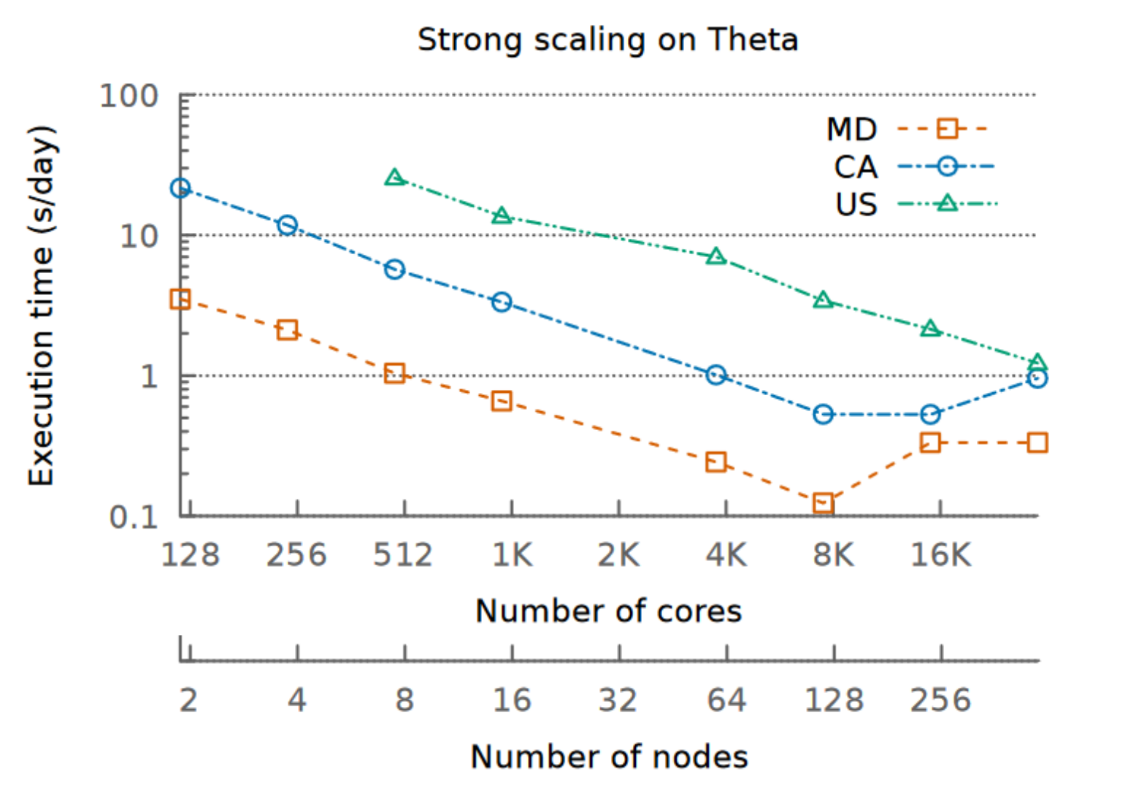
Dataset	visits	people	locations
REAL (CoC)	1,332,029	41,119	19,203
SYN (MD)	32,500,000	6,250,000	1,254,400
SYN (CA)	202,800,130	39,000,025	7,225,344
SYN (US)	1,715,829,570	329,967,225	80,281,600

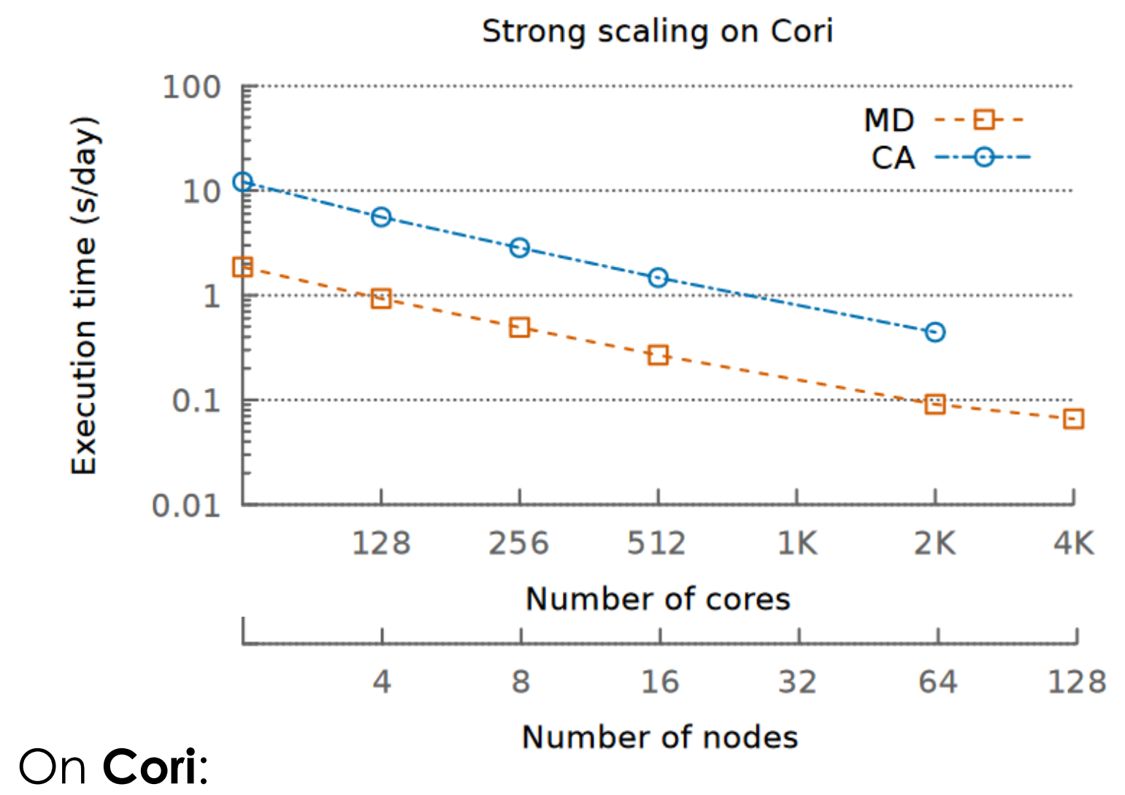
Datasets used for experiments

Results

On **Theta**:

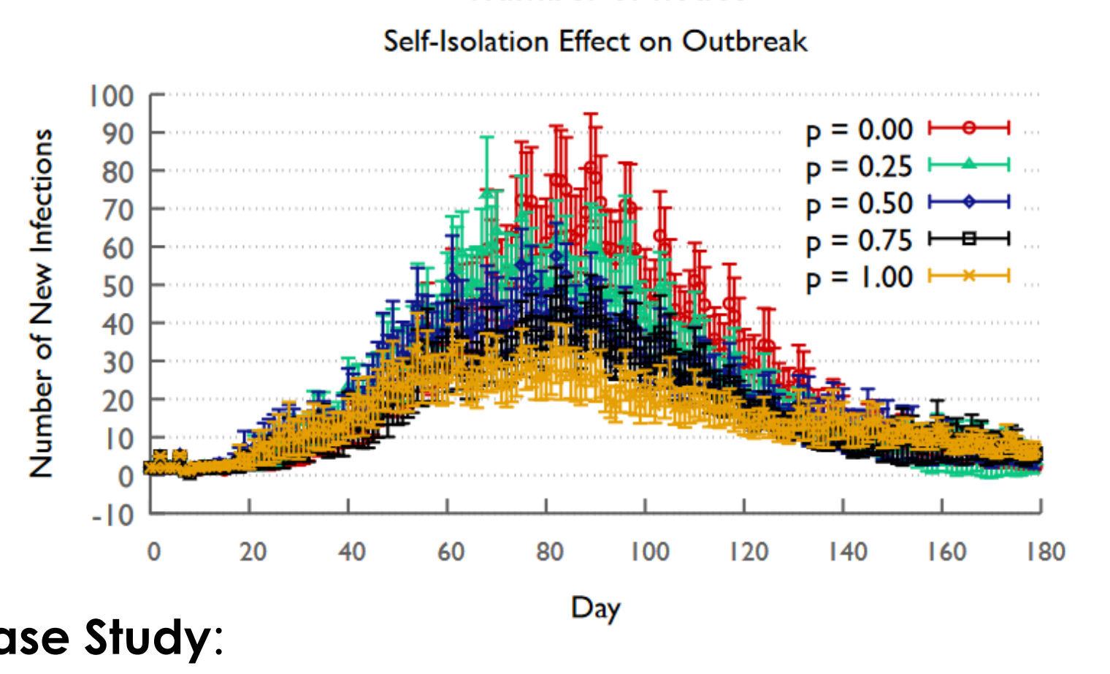
- The full-US dataset scales linearly up to 32K cores (512 nodes)
- The California and Maryland datasets only scale linearly up to 8K cores (128 nodes)
- Above 8K cores, they begin to suffer from overhead
- Loimos achieves a speedup of ~40.81 on the CA dataset when running on 8k cores





On **Cori**:

- Loimos obtains a speedup of ~28 on the CA dataset when running on 4k cores (128 nodes)



Case Study:

- Infections are reduced the more people follow the self-isolation intervention
- However, the shape of the epidemic curve does not change in our simulation

Conclusions

We present a scalable parallel simulation framework for modeling contagion processes, Loimos, and demonstrate its capabilities.

We show that Loimos can:

1. Efficiently utilize resources on the NERSC Cori and ALCF Theta machines
2. Model the impact of interventions on a population of interest

Future Work

- Switch to using real state population datasets
- Combine real state population datasets into full-US dataset
- Repeat scaling studies on realistic datasets
- Implement graph-based static load balancing
- Investigate influence of social contact graph characteristics on performance
- Implement arbitrary intervention model
- Validate simulation output against related application results
- Compare simulation output with real-world case data

Acknowledgements

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] Jones, K. E., Patel, N. G., Levy, M. A., Storeygard, A., Balk, D., Gittleman, J. L., & Daszak, P. (2008). Global trends in emerging infectious diseases. *Nature*, *451*(7181), 990–993. https://doi.org/10.1038/nature06536