DISEASE SPREAD SIMULATION-OPENMP

Sumanth Reddy Dasi (nagabrah@buffalo.edu)

Instructor: Dr. Russ Miller

CSE 708-Seminar

University at Buffalo The State University of New York

Agenda

- Disease Spread Models
- Need to Parallelize
- Scenario
- Simulation Steps
- Parallelization
- Observations

Disease Spread Model

What?

- Mathematical modelling : process of describing a real world problem in mathematical terms
- Outbreak in a population : in our case the system we are trying to model is the disease spread across individuals in a population

Disease Spread Model

When?

- Predict the spread
- Evaluate control strategies
- Real time planning

The earliest account of mathematical modelling of spread of disease was carried out in 1760 by Daniel Bernoulli.

SIR Model

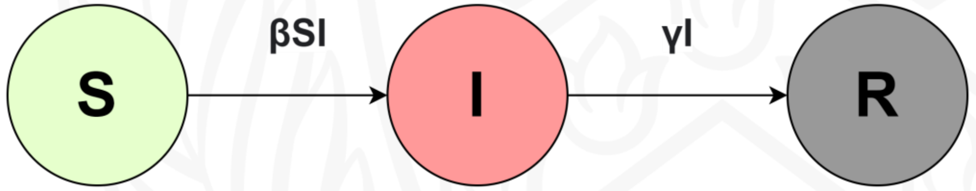
- Susceptible (S) : susceptible to the infection
- Infected (I) : infected and capable of spreading infection to others
- Removed (R) : recovered from infection and gained immunity

$$\begin{tikzcd} S \arrow[r] & I \arrow[r] & R \end{tikzcd}$$

SIR Model

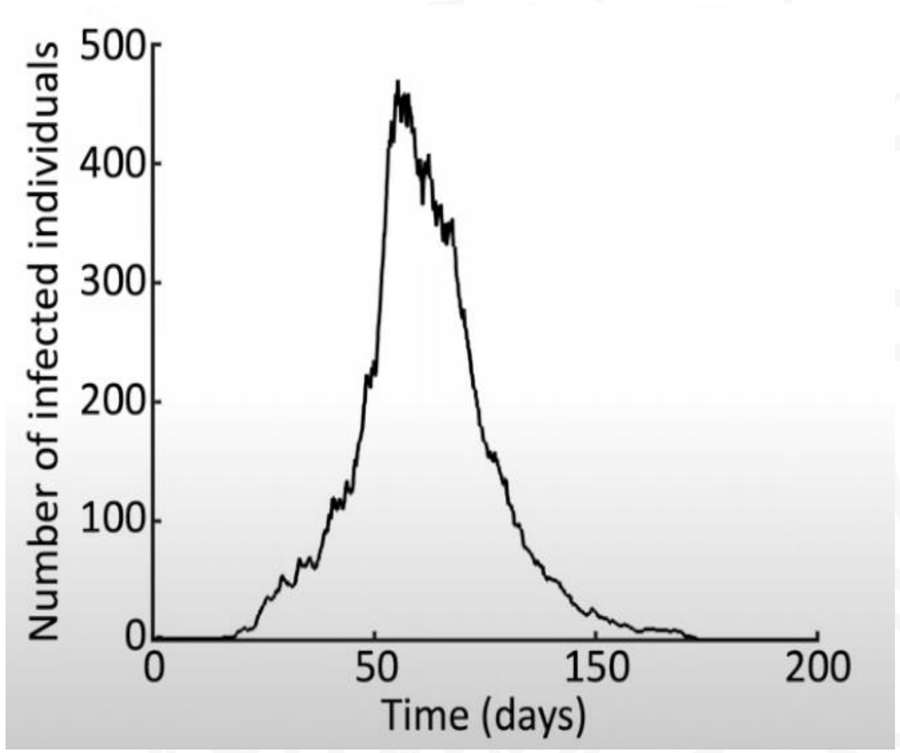
- Newly Infected : βSI
  - Proportional to S (targets)
  - Proportional to I (infected individuals)

- Newly Removed : γI
  - Proportional to I (infected individuals)


$$\begin{tikzcd} S \arrow[r, "\beta SI"] & I \arrow[r, "\gamma I"] & R \end{tikzcd}$$

Extensions to model

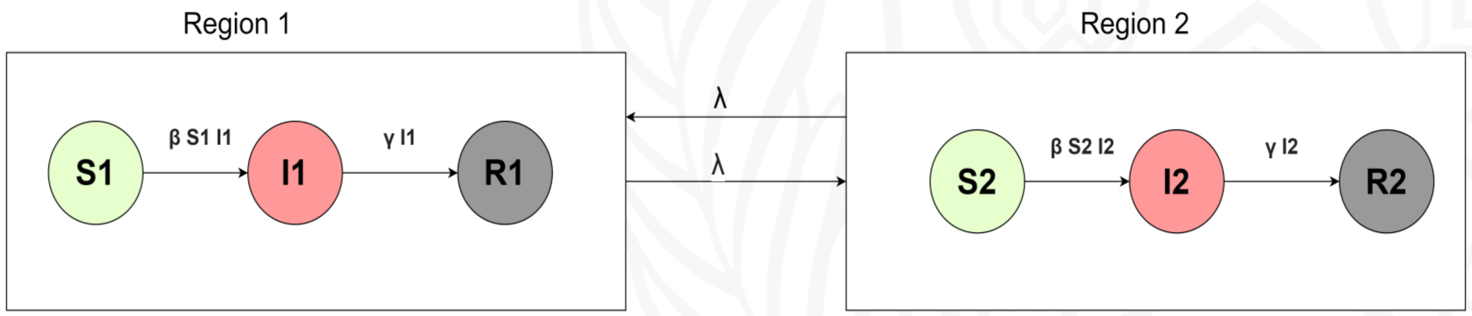
- **Stochastic model :** The initial model is deterministic, so we introduce stochasticity in infection spread to our model to make it closer to real world.



Source: Oxford Mathematics Public Lecture Robin Thompson link

Extensions to model

- Stochastic spread
- **Spatial structure :** In the real-world individuals are scattered across regions so we add this extension to the model. Moreover, different groups are differently connected with each other, so the infection moves across the groups based on how well they are connected.



- λ transfer rate

Extensions to model

- Stochastic spread
- Spatial structure
- **Infected time :** Based on the epidemiology of the disease, different infections last for different days. SIR models can be extended to include this behavior.

University at Buffalo The State University of New York

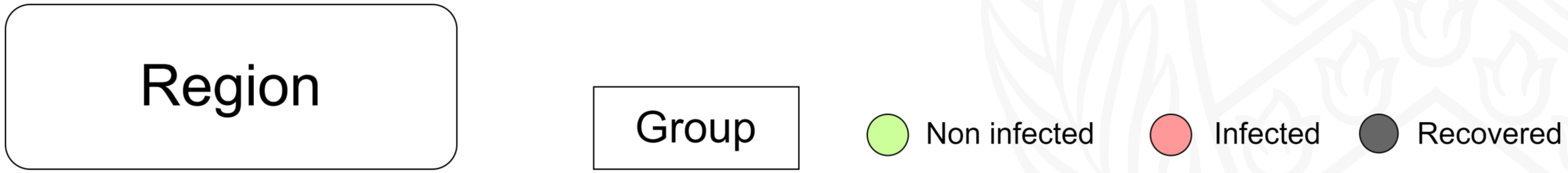
Need to Parallelize

- Faster results lead to faster safety measures
- Can help simulate scenarios with complex interactions

Simulation Scenario

- Regions
- Groups
- Individuals

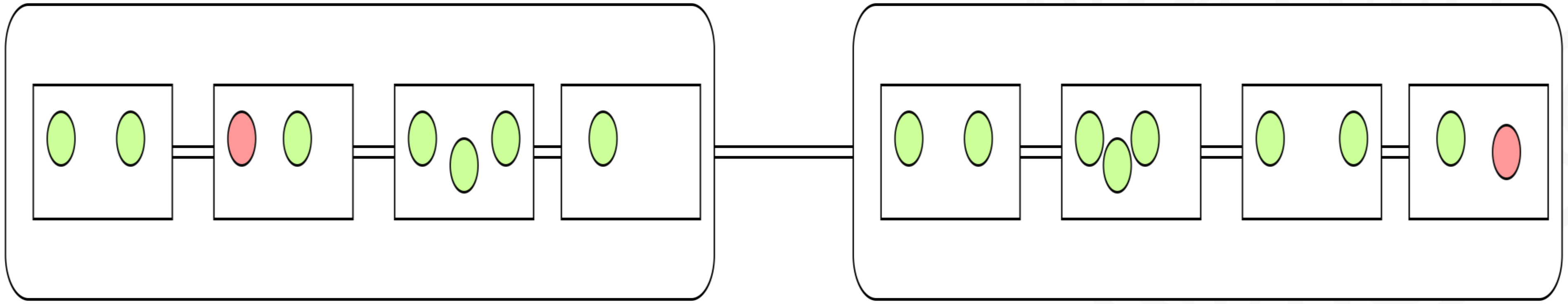
The simulation world is divided into regions which is further divided into groups. Each group has individuals in different states.



University at Buffalo The State University of New York

Simulation Scenario

- Regions
- Groups
- Individuals



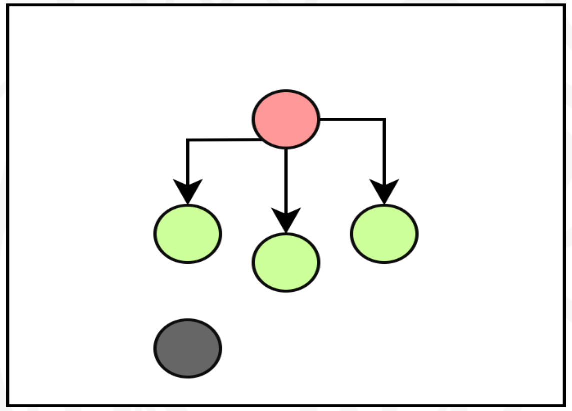
Simulation Steps

- Spread with in the group
- Spread across groups
- Caution against spread

Simulation Steps

1. **Spread with in the group:**

   In the first step, we simulate the scenario of an infected individuals spreading infection to susceptible individuals within the same group.

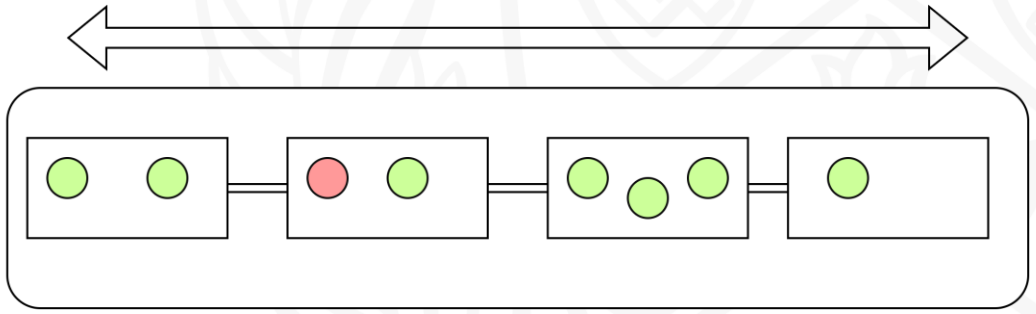


University at Buffalo The State University of New York

Simulation Steps

1. Spread with in the group
2. **Spread across groups :**

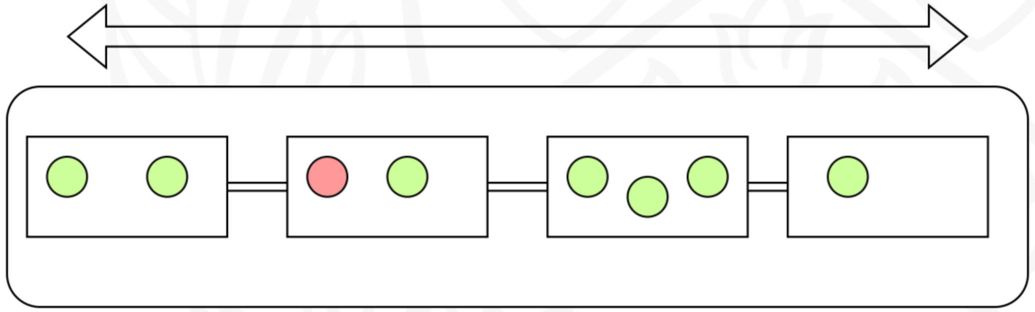
   In the second step, we simulate infected individuals moving across the groups and spreading infection in other groups.



Simulation Steps

1. Spread with in the group
2. Spread across groups
3. **Caution against spread:**

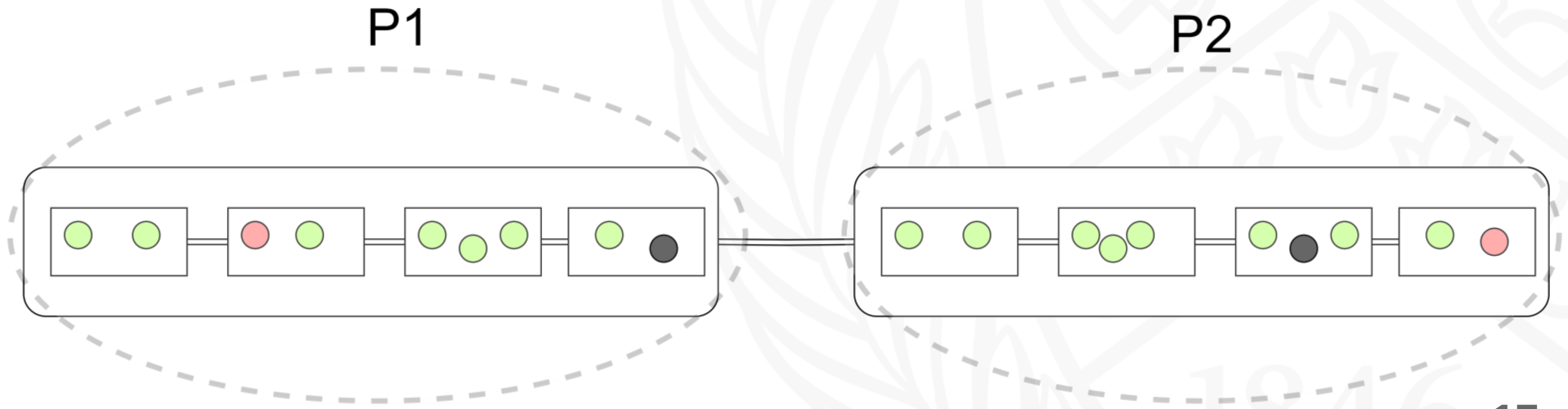
   In the third step, each region checks the number of infected individuals among its groups and if the value exceeds a certain threshold safety measures are enforced.



University at Buffalo The State University of New York

Parallelization

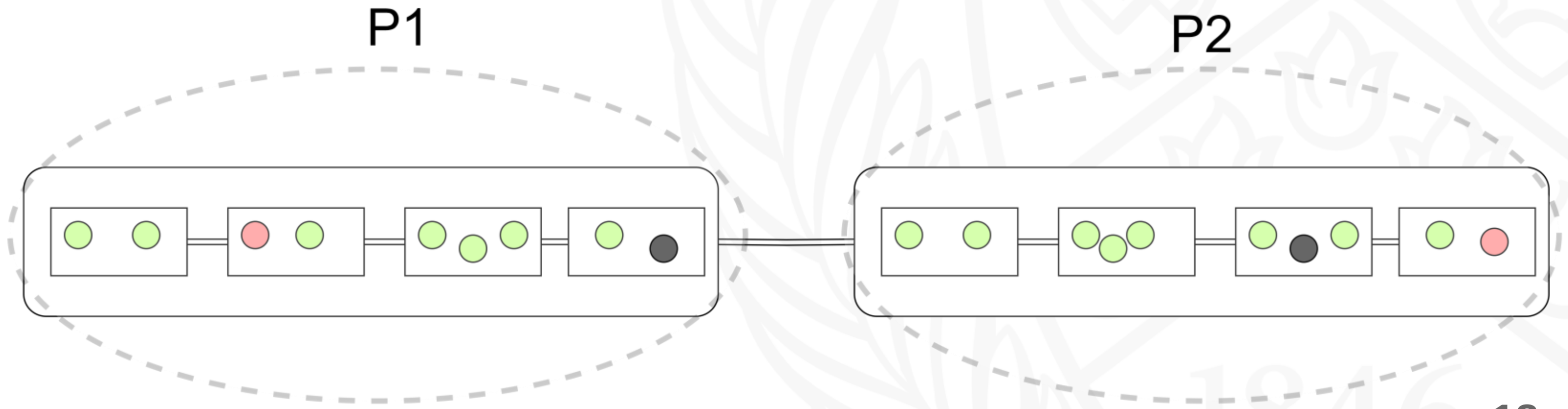
1. **Spread with in the group:**
   This step can be parallelized without any communication across Processing Elements(P.E ,threads in OpenMP) as each P.E has group information needed to simulate the step 1, so this step can be done simultaneously and in parallel across all P.E.



Parallelization

1. **Spread with in the group**
2. **Spread across groups:**

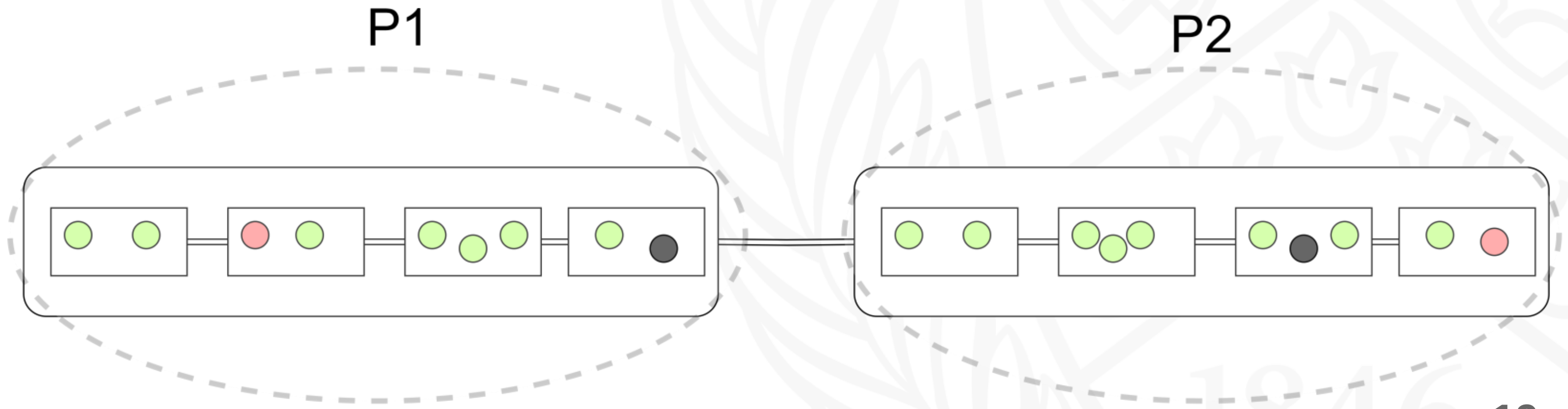
   If the infected individual travels across groups that are handled by different P.E, then P.E communicate(using shared memory) to simulate this step.



Parallelization

1. **Spread with in the group**
2. **Spread across groups**
3. **Caution against spread:**

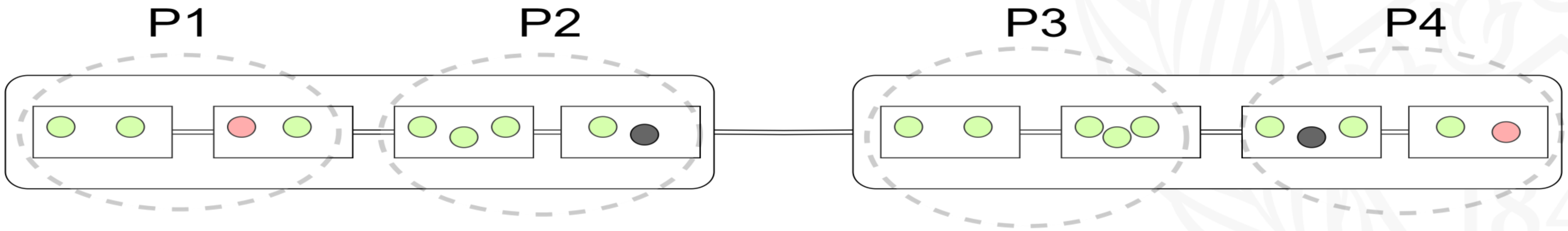
   case 1: If a P.E has all the region information in its memory(as shown in figure below by dotted oval), then it can calculate the total infected individuals in a region by itself without communicating with other P.E's.



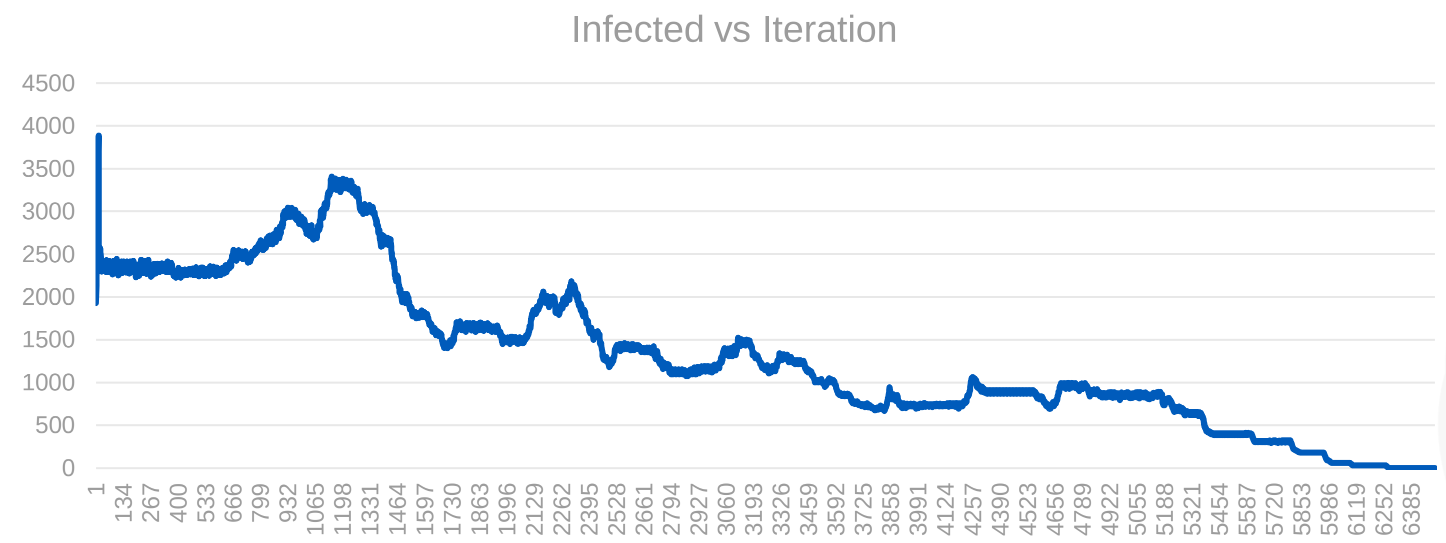
Parallelization

1. **Spread with in the group**
2. **Spread across groups**
3. **Caution against spread:**

   case 2: If data of a region is spread across multiple P.Es as shown in figure below(e.g., region 1 is spread across P1 and P2) , then data is combined using the shared memory and finally each thread knows the total infected individuals in a region.



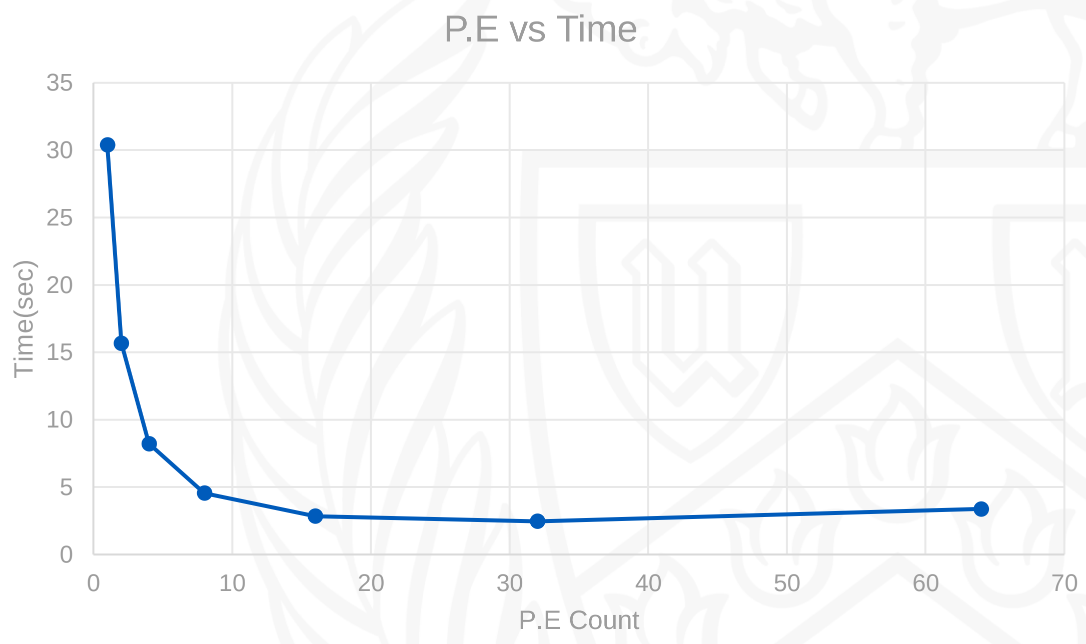
Observations: Simulation Result



The result of disease spread simulation. Initially some individuals are affected among some groups. At the start the spread is high as there are lot of susceptible individuals. Then the infected count reaches the peak and regions try to enforce safety measure among groups and reduce the infection spread. After some iterations, some individuals start recovering and the number of susceptible individuals start decreasing so the number of new infected individuals decrease as well.

P.E vs Time Taken

P.E Count	Time(sec)	Data/P.E
1	30.388506	24000
2	15.677489	12000
4	8.222139	6000
8	4.551874	3000
16	2.848532	1500
32	2.463594	750
64	3.379505	375

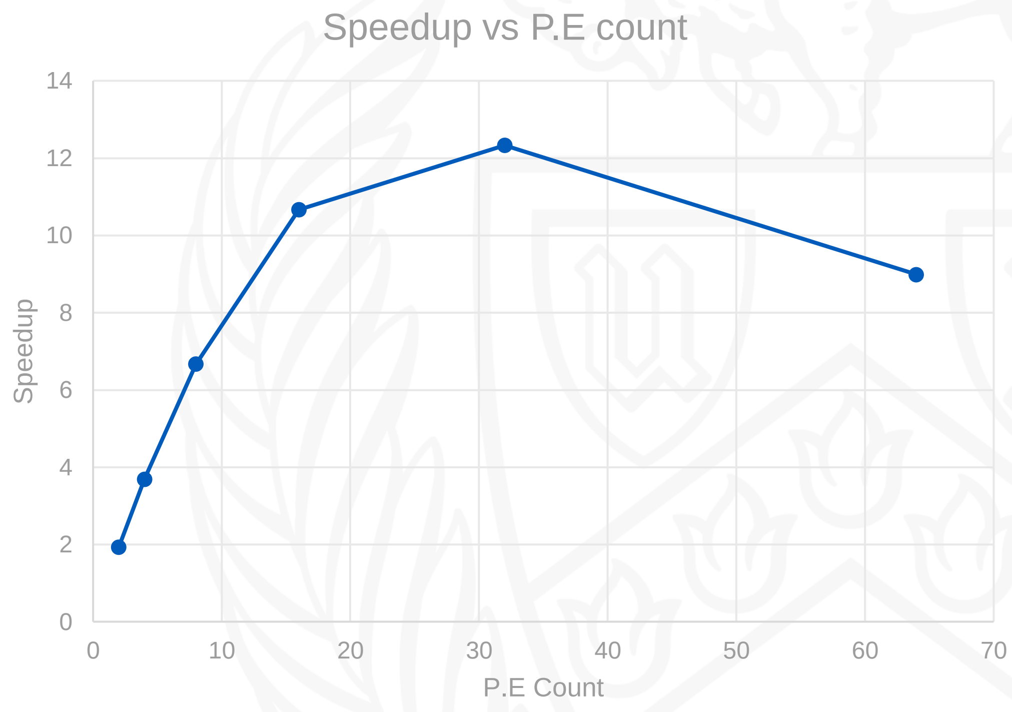


From 64 P.E we can observe communication overhead starting to be more than processing time.

Speedup

speedup = time(seq)/ time(parallel)

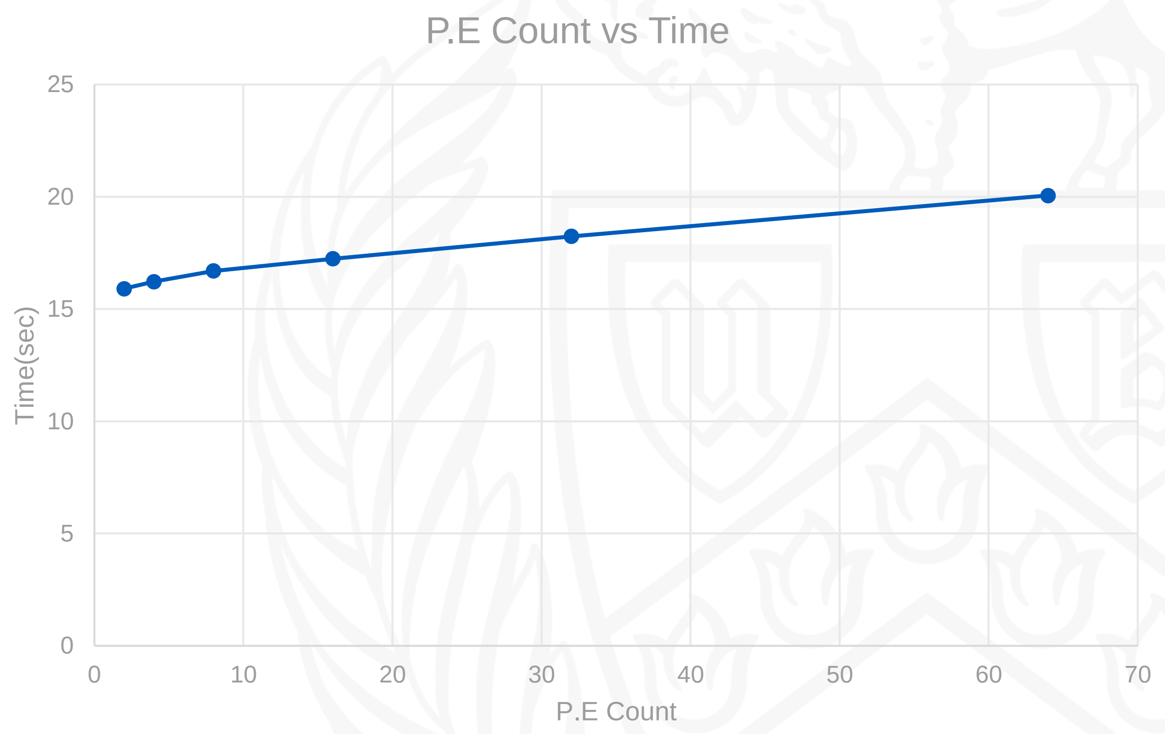
P.E Count	Speedup	Data/P.E
2	1.93832061	12000
4	3.69587525	6000
8	6.67593172	3000
16	10.6679511	1500
32	12.3348247	750
64	8.9918494	375



University at Buffalo The State University of New York

Scaling (Gustafson's law)

P.E Count	Time(sec)	Data/P.E
2	15.90657	12000
4	16.21897	12000
8	16.692461	12000
16	17.236187	12000
32	18.234454	12000
64	20.054532	12000



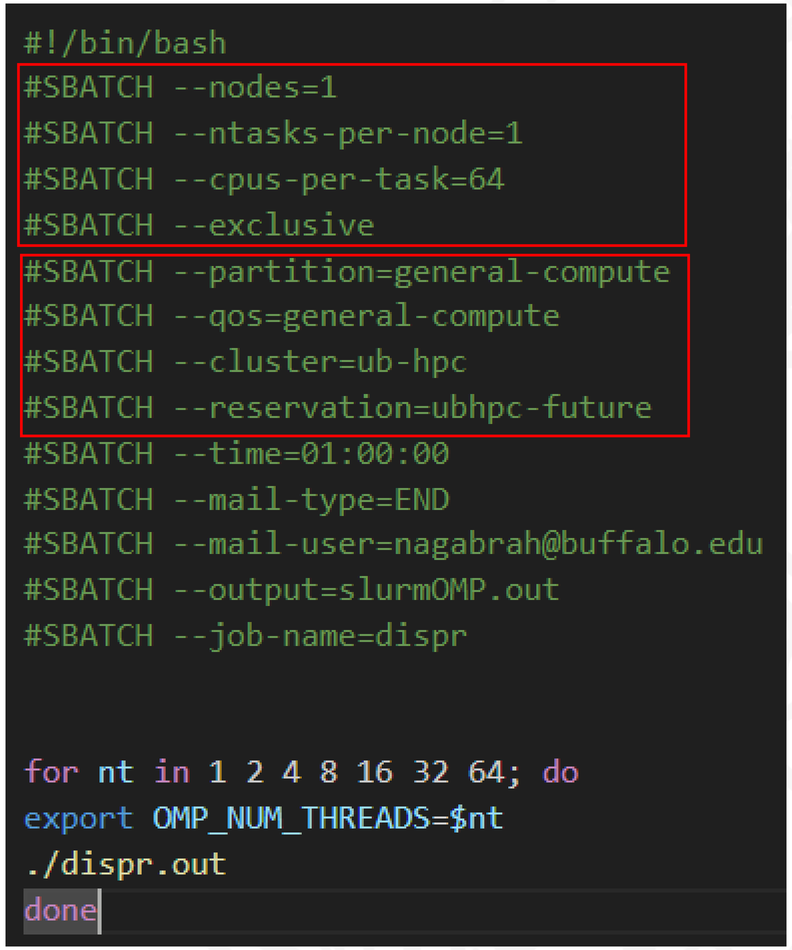
Total data for each run = 12000 * (No. of P.E)

Slurm Script

Job script is designed to run on a single node, utilize the entire node exclusively, and parallelize the computation using 64 CPU cores. Script based on Dr. Matt Jones slides on OpenMP

General computer and ubhpc-future for 64 core nodes

The OMP_NUM_THREADS environment variable sets the number of threads to use for parallel regions



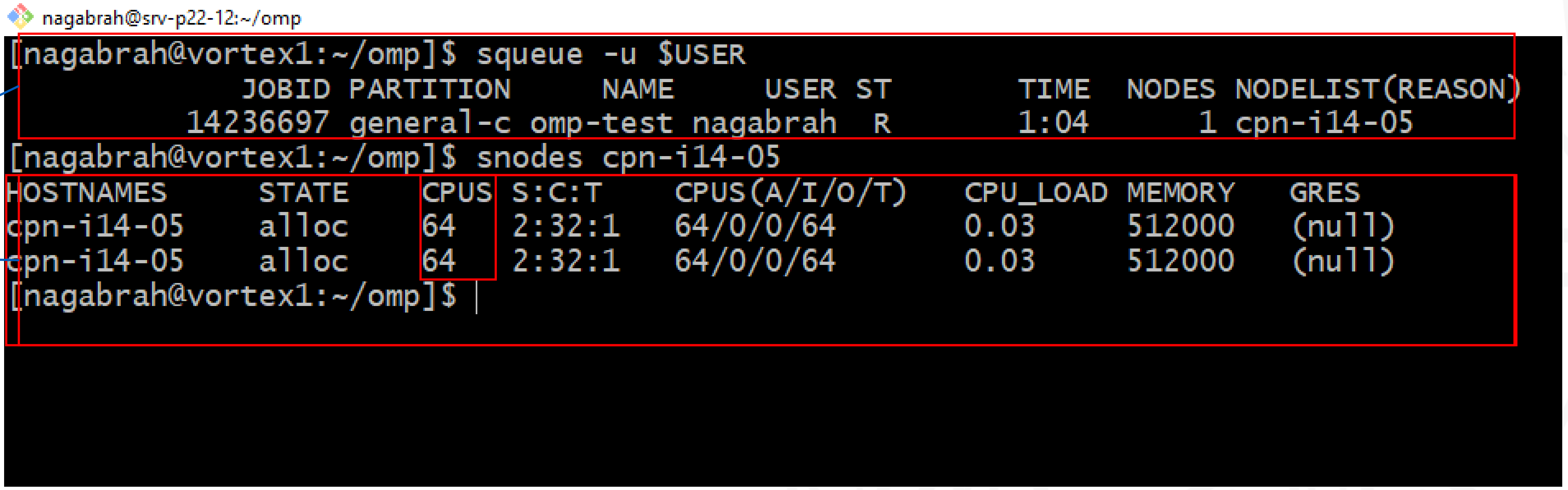
```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=64
#SBATCH --exclusive
#SBATCH --partition=general-compute
#SBATCH --qos=general-compute
#SBATCH --cluster=ub-hpc
#SBATCH --reservation=ubhpc-future
#SBATCH --time=01:00:00
#SBATCH --mail-type=END
#SBATCH --mail-user=nagabrah@buffalo.edu
#SBATCH --output=slurmOMP.out
#SBATCH --job-name=dispr


for nt in 1 2 4 8 16 32 64; do
export OMP_NUM_THREADS=$nt
./dispr.out
done
```

Run the script

Job running on node cpn-i14-05

node info of cpn-i14-05 (64 core node)



```
[nagabrah@vortex1:~/omp]$ squeue -u $USER
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          14236697 general-c omp-test nagabrah  R       1:04      1 cpn-i14-05
[nagabrah@vortex1:~/omp]$ snodes cpn-i14-05
HOSTNAMES      STATE     CPUS S:C:T     CPUS(A/I/O/T)     CPU_LOAD MEMORY     GRES
cpn-i14-05     alloc     64   2:32:1    64/0/0/64         0.03     512000     (null)
cpn-i14-05     alloc     64   2:32:1    64/0/0/64         0.03     512000     (null)
[nagabrah@vortex1:~/omp]$
```

OpenMP or MPI

- Learning Curve
- Ease of Use
- Scalability
- Industry Usage

University at Buffalo The State University of New York

References

- https://ubccr.freshdesk.com/support/solutions/articles/13000026245-tutorials-workshops-and-training-documents
- Algorithms Sequential and Parallel: A Unified Approach (Dr. Russ Miller, Dr. Laurence Boxer).
- https://docs.ccr.buffalo.edu/en/latest
- How do mathematicians model infectious disease outbreaks?
- Mathematical modelling of infectious diseases
- OpenMP Tutorials