ESCAPE

Epidemic Simulator Compiler and Programming Environment

Parantapa Bhattacharya,
Srini Venkatramanan, Bryan Lewis,
Jiangzhuo Chen, Stefan Hoops,
Henning S. Mortveit,
and Madhav Marathe

Biocomplexity Institute
University of Virginia

The ESCAPE Framework

- ESCAPE: Epidemic Simulator Compiler and Programming Environment
- A novel framework for creating agent-based epidemics simulators
- A meta-simulator for creating epidemic simulators on contact networks
- Supports multiple interacting contagions
- Aim 1: Make agent-based simulations easier to write
- Aim 2: Make agent-based simulations faster to run

ESL: Epidemic Simulation Language

- A domain specific language for describing epidemic simulations
- Syntax inspired by Python/Julia
- Comes with Editor/IDE support
  - Provides custom Language Server
  - Currently tested on NeoVim and Jupyter Lab
    - Easy to port to VSCode, Emacs, ..
- Compiles down to C++ with OpenMP / CUDA*

```
contagion c1
    state type c1_state_t

    transition
        E -> Ipresymp, p = p_E_Ipresymp, dwell = fixed3
        E -> Iasymp, p = p_E_Iasymp, dwell = normal5
        Ipresymp -> Isymp, dwell = fixed2
        Isymp -> R, dwell = isymp_r_dwell
        Iasymp -> R, dwell = normal5
    end

    transmission
        Ipresymp => S -> E
        Isymp => S -> E
        Iasymp => S -> E
    end

    susceptibility fn c1_susceptibility
    infectivity fn c1_infectivity
    transmissibility fn c1_transmissibility
    enabled fn c1_enabled
end

def p_E_Ipresymp(n: node) -> float:
    return 0.65
end

def p_E_Iasymp(n: node) -> float:
    return 0.35
end


def c1_susceptibility(v: node) -> float:
    if v.c1.state == S:
        return 1.0
    else:
        return 0.0
    end
end

def c1_infectivity(v: node) -> float:
    if v.c1.state == Isymp or v.c1.state == Iasymp:
        return 1.0
    else:
        return 0.0
    end
end

def c1_transmissibility(e: edge) -> float:
    return transmissibility_scale * e.duration / 86400
end

def c1_enabled(e: edge) -> bool:
    if (e.is_non_home_edge and
        (is_isolating(e.source_node) or is_isolating(e.target_node))):
        return False
    elif e.is_school_edge and not is_school_day():
        return False
    else:
        return True
    end
end
```

Preliminary performance results

- Test case: SIR model for whole USA
  - 330M agents
  - 12B contacts per day
  - 365 days
- Test cluster: Rivanna @ UVA
  - CPU: Intel Xeon 2 x 20 cores
  - MEMORY: 384 GB
- Digital similar:
  - 48 US States + DC
- ESCAPE init time: 55s
- ESCAPE time per tick: 3.2s

Population	Metric	EpiHiper	ESCAPE
USA	Runtime	28m 20s	11m 40s
	# Nodes	15	1
	Speedup	1	**36.43**

Preliminary performance results

- Test case: In-school NPIs [1]
  - Hybrid Learning: schools open Monday, Tuesday, and Wednesday for in-person learning; Thursday and Friday remote learning.
  - In-school testing: Daily Antigen Test on in-person school days (80% sensitivity); Weekly Antigen Test on Mondays (95% sensitivity)
  - Voluntary home isolation: Stay at home for 14 days after testing positive
  - Seeding: 5 infections every day for 10 days
- Test cluster: Anvil @ Perdue
  - CPU: AMD EPYC 2x64 cores
  - MEMORY: 256 GB
- Digital similars: CA, VA

Population	Metric	EpiHiper	ESCAPE
Virginia	Runtime	2m 4s	19.7s
	# Nodes	1	1
	Speedup	1	**6.29**
California	Runtime	6m 12s	1m 47s
	# Nodes	4	1
	Speedup	1	**13.90**

[1]: https://epihiper.readthedocs.io/en/latest/examples/examples.html

What makes the ESCAPE simulations faster?

Design: Compiled vs interpreted code

Data structure design: data oriented vs. object oriented

Compiled vs Interpreted code
Fewer instructions to execute

- How to get susceptibility for a node?
- EpiHiper approach:
  - build up data structure
  - query at runtime
- ESCAPE approach
  - Convert to code
  - Call at runtime

```
CModel {
 ...
 vector<CHealthState> mStates;
 map<string, CHealthState *> mId2State;
 ...
}
```

```
float_type _c1__susceptibility(node_index_type _v) {
        if (NODE_TABLE->_c1_state[_v] == _S) {
            return 1.0;
        }
        else {
            return 0.0;
        }
}
```

Compiled vs Interpreted code
Compile time optimizations

```
transition
    E -> Ipresymp, p = p_E_Ipresymp, dwell = fixed3
    E -> Iasymp, p = p_E_Iasymp, dwell = normal5
    Ipresymp -> Isymp, dwell = fixed2
    Isymp -> R, dwell = isymp_r_dwell
    Iasymp -> R, dwell = normal5
end
```

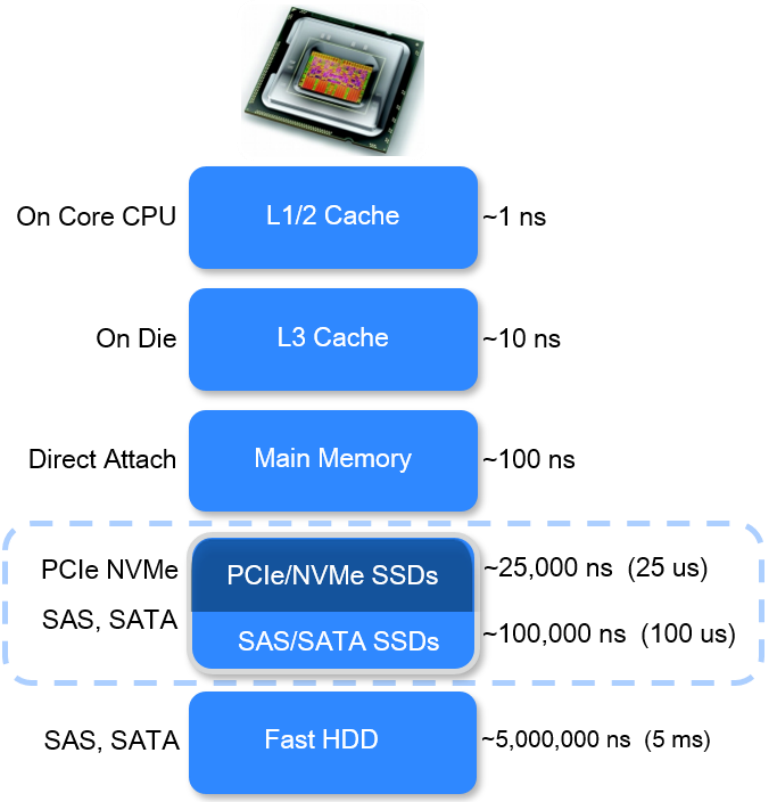
State transitions

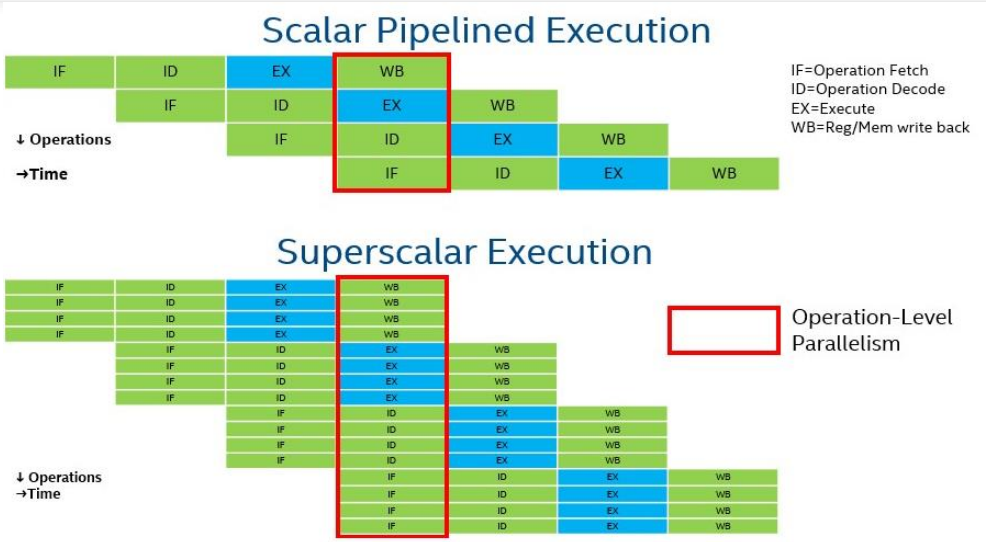
- Only state "E" has multiple exit states,
- This known at compile time
- No need to go through/generate code paths that test/check if state has multiple exit states

```
switch (state) {
case _Ipresymp:
    next_state = _Isymp;
    dwell_time = _fixed2();
    break;
case _Isymp:
    next_state = _R;
    dwell_time = _isymp_r_dwell();
    break;
case _Iasymp:
    next_state = _R;
    dwell_time = _normal5();
    break;
case _E:
    {
        float_type cum_probs[2] = {0};
        cum_probs[0] = _p_E_Ipresymp(v);
        cum_probs[1] = _p_E_Iasymp(v);
        cum_probs[1] += cum_probs[1 - 1];

        auto p = uniform01(*THREAD_RND_STATE);
        p *= cum_probs[2 - 1];

        if (p <= cum_probs[0]) {
            next_state = _Ipresymp;
            dwell_time = _fixed3();
            break;
        }
        if (p <= cum_probs[1]) {
            next_state = _Iasymp;
            dwell_time = _normal5();
            break;
        }
    }
}
```

Generated C++ code

Data oriented vs. object-oriented design: Background

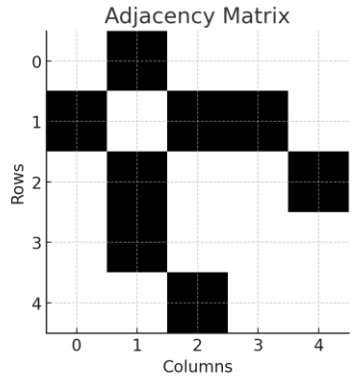


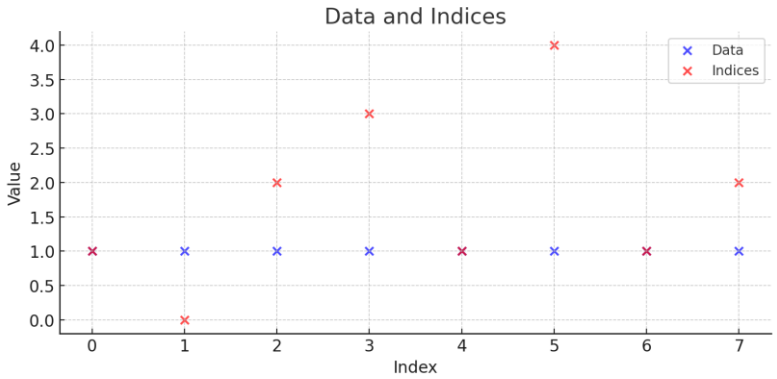


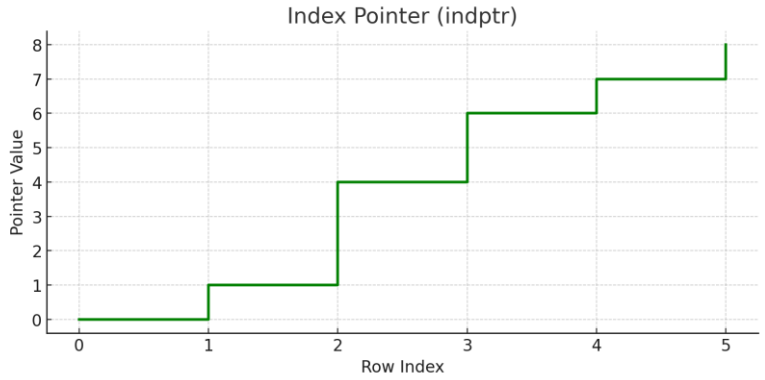
- Typical numbers
  - 1 CPU instruction: 0.25-0.5ns
  - Branch misprediction: 5ns
  - L1 cache reference: 0.5-1ns
  - L2 cache reference: 4ns
  - Main memory reference: 100ns

Data oriented vs. object-oriented design: Graph structure memory layout

- Graph data structure
  - Sparse incidence matrix
  - (<node index>, <incoming edge index>)
- Node attributes table
  - Column oriented (struct of arrays)
  - Sorted by Node ID
- Edge attributes table
  - Column oriented (struct of arrays)
  - Sorted by (target node ID, source node ID)





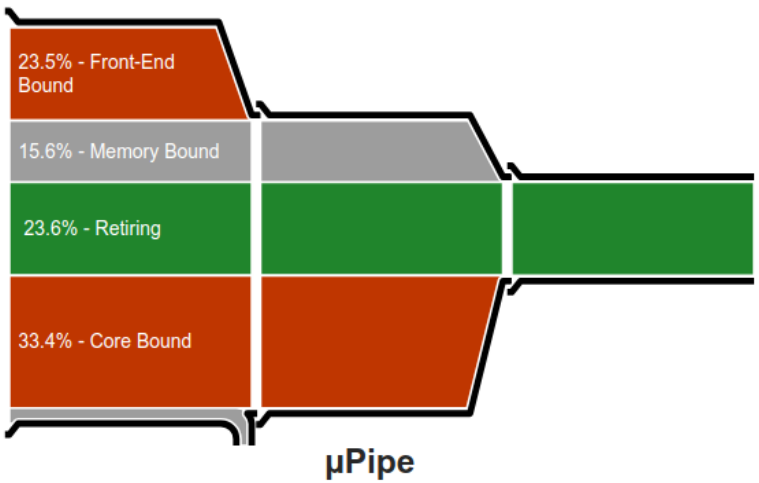


VTune: Microarchitecture exploration**

Test case: VA Montgomory; Test system: PB's Laptop; Test Scenario: Example 1 (C)

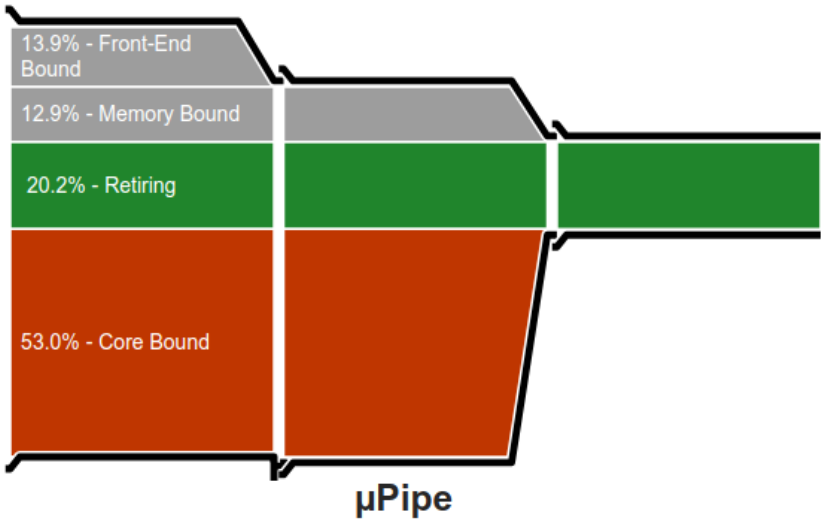
EpiHiper

- Larger frontend bound section
  - CPU waiting for instructions
  - Branch mispredictions
  - Function pointer jumps
- Slightly larger memory bound section
  - CPU waiting for data



ESCAPE

- Smaller frontend bound section
- Slightly smaller memory bound section
- Larger core bound section
  - Limited by CPU power



ESL: A DSL for writing ESCAPE simulations

- Syntax inspired by Python/Julia
- Domain specific constructs
  - Node/edge table
  - Contagion
    - Transitions/Transmissions
  - Dwell time distributions
  - Transition probability functions
  - Node/edge sets
- Serial programming constructs
  - Scalar variables: local/global (int/float/bool/...)
  - Custom enumerated types
  - Conditionals: if/elif/else
  - Loops: while/for*
  - Expressions / functions
- Parallel constructs
  - Select, Apply and Reduce

ESL: A DSL for writing ESCAPE simulations

- ESL compiles down to C++/CUDA*
- Compiler written in Python
  - Current parser uses TreeSitter
    - Supports GLR grammar
  - Grammer written in Javascript
  - Grammar compiles down to C
- 3 phase compiler
  - Source -> Parse tree
  - Parse tree -> AST
  - AST -> Generated code

- Editor support
  - Syntax highlighting
    - TreeSitter: Vim / Emacs
    - Lezer: Jupyter
  - Custom Language Server
    - Currently only error checks
    - Code completion
    - Future: code formatting
    - Vim, Emacs, VScode, Jupyter

Limitations and Future Work

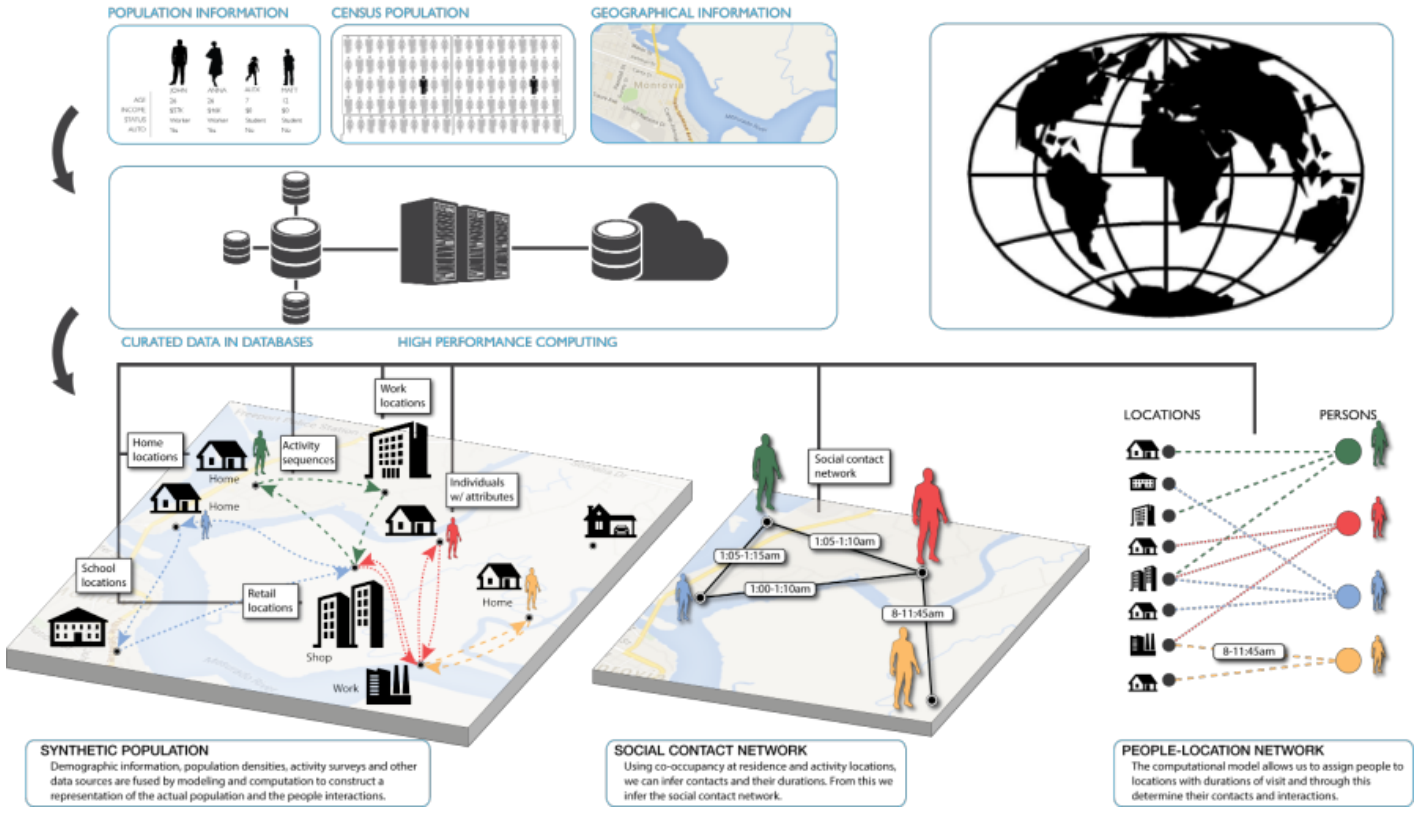
- Can work with only a finite number of contagions
- Supports only a single (possibly time-varying) contact network
- Doesn't natively support vector borne diseases
- CUDA backend is work in progress
- A distributed memory (MPI) backend implementation

`github.com/NSSAC/escape-abm`

parantapa@virginia.edu

Thank you!

Digital Similar of Populations



- ESCAPE simulators require a population to run contagions on
- The "contact network" network needs to be provided to it as an input.
- The Biocomplexity Institute maintains detailed digital similars of USA and many other countries.