# TEMPOROSPATIAL EPIDEMIC SIMULATIONS USING HETEROGENEOUS COMPUTING

**Dhananjai M. Rao**
CSE Department, Miami University
Oxford, OH 45056, USA.
email: raodm@miamiOH.edu

**KEYWORDS**
Temporospatial epidemic models, discrete event simulation, GPGPU, OpenCL, Heterogeneous Computing (HC)

**ABSTRACT**

Discrete Event Simulation (DES) is widely used for analysis of complex temporospatial epidemic models. In such simulations, a conspicuous fraction (50%–90%) of simulation runtime is typically spent in solving equations used to model epidemic progression. General Purpose Graphics Processing Units (GPGPUs) hold considerable potential to reduce time for solving epidemic equations. However, the significant differences in hardware and programming models of GPGPUs and CPUs hinder their effective use, particularly by epidemiologists and public health experts. Consequently, we have developed an epidemic modeling and simulation environment called MUSE-HC. In MUSE-HC, discrete event processing is performed on the CPU while epidemic equation processing is performed on a GPGPU. MUSE-HC provides a domain-specific modeling language called Epidemic Description Language (EDL) to streamline modeling for non-computing experts. The EDL model description is compiled and transformed to heterogeneous computing source code based on OpenCL. The generated code is compiled and executed on a workstation equipped with a GPGPU for simulation and analyses. Our experiments conducted using synthetic benchmarks show that our heterogeneous approach can improve simulation performance by up to 16× for certain temporospatial epidemic models.

## INTRODUCTION

Recent multinational epidemics of communicable and zoonotic diseases such as Zika fever, Chikungunya, influenza, etc. continue to pose serious health and socioeconomic challenges. For example, the World Bank estimates that the Zika epidemic in the Americas caused over $4 billion in losses just in 2015 with an estimated long-term cost of over $40 billion to address microcephaly cases. Hence, there is a heightened urgency to develop comprehensive methods for proactively containing epidemics. Epidemic containment strategies heavily rely on epidemiological modeling and simulation for disease forecasting and analysis. Unlike small outbreaks, multinational epidemics have complex dynamics that vary with geography and over time.

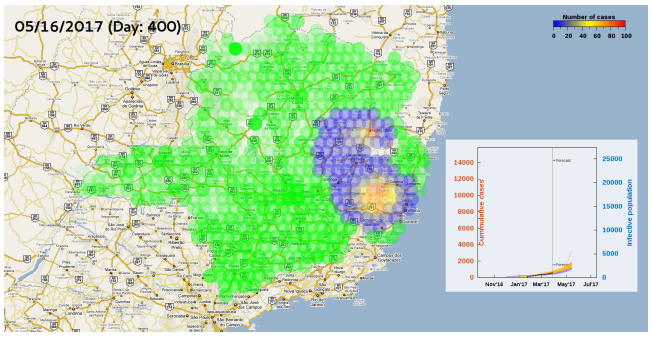
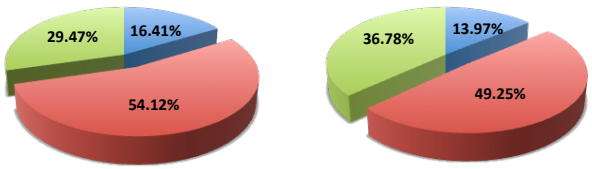



Figure 1: Example of a temporospatial epidemic model with interacting agents represented as circles (Rao et al., 2017)

Consequently, temporospatial models are used to characterize epidemic progressions. Such models are typically represented as a set of interacting "agents", where each agent models epidemic progression in a collocated population. Different approaches are used to subdivide the population resulting in a Voronoi-like tessellation. Figure 1 shows an example of a temporospatial epidemic model with circular agents that model epidemic progression in their geographic region. Interactions between agents are usually accomplished using discrete event simulation due to its advantages. Epidemic progression occurring between time steps is characterized using a system of equations. Geospatial attributes such as population, weather, etc. are incorporated as coefficients in the equations. Each agent uses the same set of epidemic equations associated with a given disease but with different coefficients corresponding to their geography.

### Computational characteristics of epidemic simulations

Computational methods for simulating epidemic progression use two key approaches – ❶ a system of Ordinary Differential Equations (ODEs) along with numerical methods are used for deterministic simulations; and ❷ a system of rate-based equations and a Stochastic Simulation Algorithm (SSA) is used for probabilistic or stochastic simulations. One of the aforementioned approaches is chosen based on analysis needs. Immaterial of the method used, a conspicuous portion of simulation-runtime is spent on solving equations. The charts in Figure 2 illustrate such an example, where nearly 50% of the simulation runtime (in red color) is spent in solving equations. The data was collected using the temporospa-




(a) Deterministic (ODE) (b) Stochastic (SSA)

Figure 2: Proportion of simulation runtime in different operations collated using Linux `perf` – ■: Model processing, ■: Epidemic equations processing, ■: Infrastructure

tial model from Rao et al. (2017) with 27,000 agents. The epidemic equations for each agent were solved using a time step of 0.05, *i.e.,* 20 steps per day. The ODE version uses Runge-Kutta 4th order method while the SSA version uses Gillespie's algorithm with Tau+Leap optimization.

### Long run times of epidemic simulations

The computational time for solving epidemic equations grows polynomially, often to $\gg 90\%$, with changes to different settings, including: decrease in step size (often 0.01 or 0.005 is used), increase in model size or number of agents, increase in complexity of equations, etc. The computational overheads are further magnified by the need to run 1000s of simulation replications for various analyses. Consequently, reducing runtime of such epidemic simulations is important.

### Heterogeneous computing: Opportunities & challenges

Heterogeneous Computing (HC) involves the use of more than one type of processor for running a program. Typically, a combination of standard CPU and a General Purpose Graphics Processing Unit (GPGPU) is used for HC. Modern GPGPUs can deliver over a teraflop of compute power when used effectively. A GPGPU is essentially a massive Single Instruction Multiple Data (SIMD) processor in which a single operation is simultaneously performed on thousands of data items using independent compute units. Consequently, GPGPU programming follows a different paradigm and requires the use of different software systems such as CUDA or OpenCL.

The GPGPU architecture is conducive for epidemic simulation because all the agents in a temporospatial model use the same set of equations (*i.e.,* single set of instructions) but with different coefficients (*i.e.,* multiple data). Consequently, unlike a CPU, a GPGPU can process thousands of equations simultaneously. On the other hand, modern CPUs are highly optimized for logic and conditional operations which play a dominant role in discrete event simulations. Consequently, heterogeneous computing performed using CPU and GPGPU holds considerable promise to reduce simulation times. Reduction in simulation time is critical in enabling comprehensive epidemic analyses that often require thousands of simulation replications.

### Motivation & overview of proposed work

Realizing the computational advantages of heterogeneous computing requires considerable technical skill and development of software for different programming paradigms. The issues are compounded by the need to develop both ODE and SSA versions. Moreover, changes to equations require consistent modifications to various versions. These issues hinder effective use of heterogeneous computing, particularly by the end-users, including epidemiologists and public health experts. Accordingly, to ease the effective use of heterogeneous computing (HC) for epidemic simulations, this paper presents a novel modeling and simulation environment called MUSE-HC. MUSE-HC provides an intuitive, domain-specific modeling language called Epidemic Description Language (EDL) to streamline modeling. A single EDL description is compiled and transformed to generate both ODE and SSA simulations capable of utilizing heterogeneous compute platforms. Specifically, the simulations are designed to perform discrete event processing on the CPU and epidemic equation processing on a GPGPU. The architecture, design, and use of EDL and MUSE-HC are discussed in detail. The verification and validation experiments discussed in this paper show that the results from MUSE-HC are consistent with their purely CPU-based counterparts. The performance experiments conducted using synthetic benchmarks show that heterogeneous computing than improve simulation performance by 16× for temporospatial epidemic simulations.

## BACKGROUND

The proposed transdisciplinary research aims to provide a comprehensive solution for temporospatial epidemic modeling and simulation using heterogeneous computing. Consequently, the proposed software system integrates three different disciplines, namely: ❶ temporospatial epidemic modeling, ❷ discrete event simulation using a framework called MUSE, and ❸ heterogeneous computing via OpenCL. The pertinent concepts from these three distinct disciplines are discussed in the following subsections.

### Temporospatial epidemic modeling & simulation

Computational epidemiology builds on the core concepts of modeling and simulation to enable comprehensive understanding of epidemics required for design and administration of strategies to contain them. The de facto standard for conceptual modeling of epidemics are compartmental models. Figure 3 shows a classic SEIR model in which the population is divided into independent subsets or *compartments*, namely susceptible (S), exposed (E), infectious (I), and recovered (R).

The compartments and transitions in a model are chosen based on the nature of the disease. Transitions between the compartments are governed by ecological and epidemiological parameters, such as $\alpha$, $\beta$, and $\gamma$ show in Figure 3. However, ecological characteristics can considerably vary based

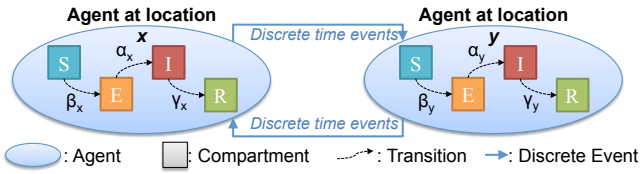



Figure 3: Example of a temporospatial epidemic model

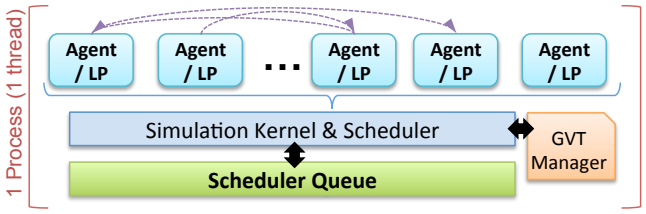



Figure 4: Overview of a MUSEsequential simulation

on geography, population density, etc. Consequently, temporospatial models further subdivide the population into independent but interacting entities or "agents" as shown in Figure 3. Each agent models epidemic characteristics of a given location (say $x$), by suitably modulating epidemic parameters (e.g., $\alpha$ to $\alpha_x$) based on ecology, demographics, weather, and other environmental factors. In addition, agents also interact with each other to model epidemic propagation and other natural processes. The inter-agent interactions are modeled through the exchange of discrete time events to enable the agents to operate independently. Figure 1 shows such a temporospatial epidemic model consisting of many interacting agents.

Progression of epidemics within an agent is simulated using one of following two methods:

1. Deterministic methods: Deterministic methods use a system of Ordinary Differential Equation (ODEs) to represent epidemic transitions. The system of ODEs is solved using numerical methods such as Runge-Kutta 4th order method.
2. Stochastic methods: Epidemics seldom progress in a deterministic manner. Stochastic methods aim to provide a more realistic simulation by characterizing the natural randomness inherent in epidemics. For stochastic simulation, the epidemic transitions are represented as a rate-based system of equations. Simulations are performed using a Stochastic Simulation Algorithm (SSA) such as Gillespie's SSA along with Tau+Leap optimization.

These methods require small time steps, e.g., 0.01 or smaller, to yield sufficiently accurate results, particularly at inflection points. The time step has a conspicuous impact on computational time, with smaller time steps requiring longer computational times. Furthermore, simulation-based analyses require 1000s of simulations to be conducted which dramatically increase analysis time.

### Miami University Simulation Environment (MUSE)

The proposed heterogeneous computing (HC) capable simulation system has been developed by significantly enhancing a general purpose, discrete event simulator called MUSE. It has been developed in C++ and exposes an object-oriented Application Program Interface (API) for modeling. A conceptual overview of a MUSE simulation is shown in Figure 4. A MUSE simulation is organized as a set of Logical Processes (LPs) or "agents" that interact with each other by exchanging virtual timestamped events. We have extended this capability to enable an LP to operate in HC mode. The `simulation kernel` shown in Figure 4 implements core functionality associated with LP registration, event processing, state saving, synchronization, and Global Virtual Time (GVT) based garbage collection. In this study, we have extended MUSE's kernel to operate in HC mode using OpenCL. In HC mode, simulations on the CPU proceeds sequentially and concurrently (all LPs are at the same simulation-time) on the GPGPU. MUSE can be downloaded from `http://pc2lab.cec.miamiOH.edu/muse`.

### OpenCL concepts & terminology

OpenCL is an open, royalty-free standard for parallel programming of computational devices, including: GPGPUs, CPUs, DSPs, and FPGAs. Unlike its competitor CUDA (which is currently supported only by NVIDIA) OpenCL is supported by many leading vendors including: Intel, AMD, IBM, Apple, Samsung, NVIDIA, ARM, and Qualcomm, to name a few. An OpenCL program consists of many `work items` called a "workgroup" running in parallel on one or more `processing elements` as shown in Figure 5. `work items` are executed using the Single Instruction Multiple Data (SIMD) paradigm. In this research, each `work item` is used to run epidemic equations for an LP or agent. A set of `work items` have access to memory organized hierarchically, which local memory being much faster than global memory. Consequently, MUSE-HC is designed to primarily work with local memory and transparently manages copying data to global memory and to CPU's main memory as neeed.

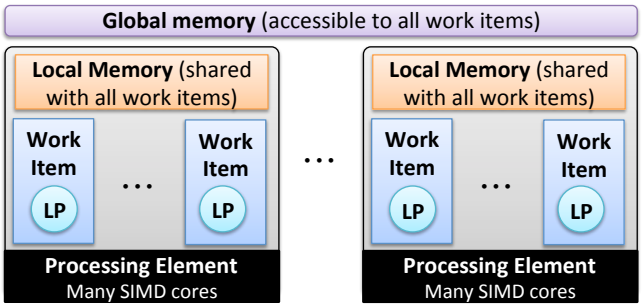



Figure 5: Overview of OpenCL runtime

## RELATED WORKS

The primary focus of this study is modeling and accelerating the simulation of temporospatial epidemic models using Heterogeneous Computing (HC). Several prior investigations have focused on parallel epidemic simulations, including works by Giridharan and Rao (2016), Rao (2016), and Yeom et al. (2014). These investigations focus on parallel epidemiological simulations using just CPUs. On the other hand, this investigation focuses on using both CPUs and GPGPUs. Consequently, this section compares and contrasts the proposed work with closely related research that involve heterogeneous computing.

Leonenko et al. (2015) propose the use of heterogeneous computing for epidemic simulations. Their work is similar to ours in that discrete event processing is performed on CPU while epidemic propagation is performed on GPGPUs. Their work focus on predefined models written using MATLAB. The use of multiple GPGPUs for epidemic simulation using NVIDIA CUDA is proposed by Shekh et al. (2015). They focus on simulating predefined SEIR individual-based models with contacts arising at buildings or homes. Epidemic simulations using multiple GPGPUs and CUDA in a cluster computing environment has also been investigated by Zou et al. (2013). Their work focuses on simulating predefined contact networks while compensating for communication latencies between the compute nodes on the cluster. Arlindo et al. (2015) explore the use of GPGPUs and CUDA for epidemic simulations. They represent each individual as a string and simulate a predefined epidemic with a fixed set of compartments.

Similar to some of the aforementioned investigations, this work also focuses on simulation of epidemics using heterogeneous computing. However, several novel aspects distinguish this research from prior investigations, namely: ❶ unlike prior investigations that use a fixed model, this study applies to any epidemic model; ❷ a novel, domain-specific modeling language called EDL is proposed in this study, while prior investigations use predefined models; ❸ a key distinction in our study is the use of temporospatial models rather than contact networks used by other investigators; ❹ in contrast to earlier studies that use a hard-coded source code, in this study the CPU and GPGPU source codes are automatically generated from EDL; ❺ in this study we have used OpenCL that is broadly usable and not CUDA that is limited to running only on NVIDIA hardware.

## MUSE-HC: ARCHITECTURE & DESIGN

Harnessing the heterogeneous computing (HC) capabilities of GPGPUs and CPUs for temporospatial modeling and simulation of epidemics requires considerable knowledge and technical skill. The programming paradigms of CPUs and GPGPUs are substantially different requiring completely different software. Moreover, changes to epidemic transitions require tedious modifications to both deterministic and stochastic versions complicating software development, troubleshooting, and maintenance. These issues hinder effective use of heterogeneous computing.

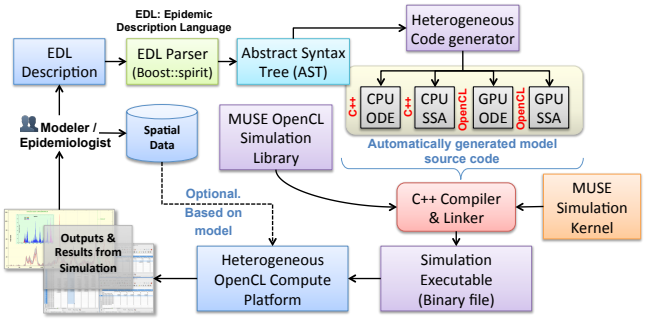



Figure 6: Process of modeling & simulation with MUSE-HC

Accordingly, to ease the effective use of heterogeneous computing (HC) for epidemic simulations, we have developed a novel modeling and simulation environment called MUSE-HC. Figure 6 illustrates the process of modeling and simulation using MUSE-HC. The primary input is a description of the compartmental model described using a domain-specific, Epidemic Description Language (EDL). EDL has been designed to provide an intuitive, yet expressive constructs for describing compartmental models as discussed in the following subsection. The EDL description is parsed into an intermediate Abstract Syntax Tree (AST) using a custom parser. The EDL parser has been developed in C++ using `BOOST`'s `Spirit` library. The AST is used by an a code generation module that converts the AST into semantically equivalent versions of source codes for conducting both deterministic and stochastic simulations on CPU and GPGPU. Note that a single EDL description is compiled and transformed to generate both ODE and SSA simulations capable of utilizing heterogeneous compute platforms. A single, intuitive source minimizes overheads associated with model development, validation, and maintenance.

The generated model is then complied and linked with the MUSE-HC simulation kernel to produce the final executable. The simulation kernel provides generic functionality for discrete event simulation and GPGPU-based operations. The MUSE-HC simulations are designed to perform discrete event processing on the CPU and epidemic equation processing on a GPGPU. The kernel also generates results in the form of Comma Separated Values (CSV) files to ease visualization and further analyses. MUSE-HC is also discussed in detail in this section.

### Epidemic Description Language (EDL)

The Epidemic Description Language (EDL) has been designed to ease effective use of heterogeneous computing (HC) by subject-matter experts including epidemiologists and public health practitioners. EDL provides intuitive, domain-specific language constructs that ease description of compartmental models in a machine processible format. For example, consider the Ebola compartmental model shown

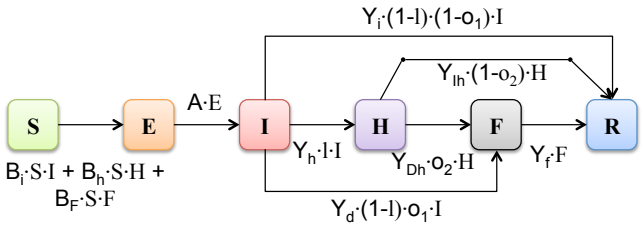



Figure 7: Compartmental model for Ebola proposed by Legrand et al. (2007) and used by Rivers et al. (2014)

in Figure 7 that was originally proposed by Legrand et al. (2007). The conceptual model has 6 compartments: Susceptible ($S$), Exposed ($E$), Infectious ($I$), Hospitalized ($F$), and Funeral ($F$). Transition of individuals between the compartments is governed by the equations shown with each transition. The equations involve several epidemiological constants that vary for different countries as discussed in detail by Legrand et al. (2007) and Rivers et al. (2014).

The EDL description for the compartmental model in Figure 7 is shown in Listing 1. The description begins with the name of the epidemic (e.g., `Ebola_Liberia`) followed by four code blocks, namely `constants`, `parameters`, `compartments`, and `transitions`. The `constants` block contains named constants used to describe compartment transitions. These constants directly reflect those used in the conceptual compartmental model. The `parameters` block contains constants that are computed (*i.e.,* determined at runtime) or whose values vary with time. These include values that vary with geography, temperature, or other model characteristics. Parameters may also be used for values that are set during simulation and are used in transition equations.

```
# EDL description for the Ebola model
epidemic Ebola_Liberia {
  # Epidemic constants in this model
  constants {
    Bi = 0.16;    # Contact rate, community
    A  = 0.083;   # Incubation rate (12 days)
    Yi = 0.0667; # Infection duration (15 days)
    # More constants not shown for brevity
  }
  parameters {
     # No time-varying parameters in this model.
  }
  compartments {
     # Comments removed for brevity
     s, e, i, h, f, r;
  }
  transitions {
     e += -1, i += 1 @ (A * e);
     i += -1, h += 1 @ (Yh * l * i);
     h += -1, f += 1 @ (Ydh * O2 * h);
     f += -1, r += 1 @ (Yf * f);
     # More transitions removed for brevity
  }
}
```

Listing 1: Partial EDL description for model in Figure 7

The `compartments` code block lists all of the compartments from the conceptual model. The last code block in an EDL description defines transitions between the compartments. Each transition consists of two parts, namely: ① a comma-separated list of population changes to a subset of compartments followed by ② the transition rate equation. These two parts are separated by an `@` symbol read as "at the rate of". For example the $E \xrightarrow{A \cdot E} I$ transition from Figure 7 is directly encoded as: `e -= 1, i += 1 @ (A * e)` as shown in Listing 1. Since EDL syntax has been designed to provide a direct mapping from a conceptual model, EDL is more intuitive for use by epidemiologists and public health experts.

### EDL parser and code generator

An EDL description of an epidemic is parsed into an in-memory Abstract Syntax Tree (AST) using the EDL parser as illustrated by Figure 6. The parser has been developed in C++ using `BOOST Spirit` library (version 2.53). The `Spirit` library uses expression templates to implement an EBNF like sub-language within C++. The resulting EDL parser produced via `Spirit` is an LL(1) parser that does not require any backtracking due to the straightforward grammar for EDL. The parser performs both syntactic and semantic checks on the input EDL. The output from the EDL parser is an Abstract Syntax Tree (AST). AST is an in-memory data structure that stores the input EDL in a form conducive for further processing.

The AST produced by the EDL parser is used to generate source code for a complete model compliant with MUSE-HC's API. The primary generated artifact is an *agent* class that includes both deterministic and stochastic versions of the model. The deterministic version uses Ordinary Differential Equations (ODEs) to characterize state transitions. The ODEs are automatically generated by appropriately combining various transitions for each compartment. Listing 2 shows a snippet of the generated source code with ODEs for the Ebola EDL from Listing 1. The code in Listing 2 shows implementation for the `ode` method (used by MUSE-HC) that would typically be manually coded by a modeler. The source code is in C so that the same code can be run both on the CPU as well as on a GPGPU via OpenCL. The code uses a custom `real` user-defined type that is mapped to either the `float` or the `double` primitive type, depending on the application's needs. Similarly, source code is also generated for running stochastic versions of the model.

The EDL code generator also produces a top-level simulation C++ `class` compliant with MUSE-HC's Application Program Interface (API). This class provides command-line arguments to set initial values for different compartments in the model. In addition, it provides options for generating a grid of agents in a temporospatial model as shown earlier in Figure 1. The modeler may further enhance or modify the generated source code for fine tuning the simulation as needed. The generated source code is compiled and linked with MUSE-HC to produce the final simulation executable. The resulting executable utilizes the MUSE-HC library to enable simulating the model on heterogeneous computing capable hardware platforms.

```
typedef double real;
void ode(const int numVars, const real params[],
         const real comp[], real deltas[]) {
    const real N = (comp[s] + comp[e] + comp[i] +
       comp[h] + comp[f] + comp[r]);
    deltas[s] = (-1 * ((Bi * comp[s] * comp[i]) +
       (Bh * comp[s] * comp[h]) +
       (Bf * comp[s] * comp[f]))  / N);
    deltas[e] = (1 *  ((Bi * comp[s] * comp[i]) +
       (Bh * comp[s] * comp[h]) + (Bf * comp[s] *
       comp[f])) / N) + (-1 * (A * comp[e]));
    deltas[h] = (1 *  (Yh * l * comp[i])) + (-1 *
       (Ydh * O2 * comp[h])) + (-1 *  (Yih *
       (1 - O2 ) * comp[h]));
    // Additional equation not shown for brevity
}
```

Listing 2: Snippet of Ordinary Differential Equations (ODEs) in the C++ source code generated for EDL from Listing 1

### The MUSE-HC simulation framework

The heterogeneous computing (HC) capable simulations have been enabled by significantly enhancing a discrete event simulation framework called MUSE. The resulting simulation framework is called MUSE-HC. It has been developed in C++ using its object-oriented capabilities. The API of MUSE-HC includes a `HCAgent` class that must be implemented to perform various model related operations on both the CPU and GPGPU. The EDL code generator discussed earlier essentially generates an implementation for this class. The `HCAgent` provides a set of macros to facilitate a single version of the source to be used both on the CPU and GPGPU. Specifically, these macros convert the source code into strings. The strings are appropriately concatenated together to produce the OpenCL source code for execution on a GPGPU. The `HCAgent` class also provides a library of algorithms for deterministic and stochastic simulations, including: Runge-Kutta 4th order method, Gillespie's exact Stochastic Simulation Algorithm (SSA), Gillespie's SSA with Tau+Leap optimization, and Mersenne-Twister random number generators.

In conjunction with the `HCAgent`, MUSE-HC also requires implementation for an `HCState` classes that encapsulates the state of an agent. The state includes values for the different compartments in the model along with any time-dependent parameters. The EDL code generator also generates an implementation for the `HCState` class using the macros provided by MUSE-HC. The state is copied to-and-from the GPGPU for heterogeneous computing.

*Core simulation and overlapped execution*

The salient operations performed by MUSE-HC to run an heterogeneous computing (HC) capable simulation is summarized in Algorithm 1. Upon starting, as part of initialization, MUSE-HC first prepares the OpenCL kernel for GPGPU-based execution. The OpenCL kernel is created by combining standard MUSE-HC support library along with an agent's custom ODEs and SSA source codes. Next, the simulation proceeds in cycles in which each agent processes its next set of events in chronological order. After processing events, each agent indicates if additional HC operation is desired.

Agents requesting HC operations are tracked in an `ocl` list and scheduled for execution in batches as shown in Algorithm 1. The size of each batch is determined by a given `workgroupSize` value. The size of the workgroup plays an important role in effectively utilizing the computational resources of a GPGPU. On the GPGPU, the MUSE-HC libraries work in concert with the generated code to perform deterministic or stochastic simulation operations. The data for parameters and compartments for each agent are copied to the GPGPU to ensure fast operation. Note that data transfer is an overhead associated with all GPGPU-based computing solutions.

*Overlapped execution on CPU & GPGPU*

In MUSE-HC, executions on the GPGPU are overlapped with discrete event processing on the CPU as summarized in Algorithm 1. Specifically, the OpenCL kernels are scheduled to execute on the GPGPU in batches and the CPU continues with discrete event processing. Overlapped execution enables amortization of the overheads associated with heterogeneous computing. Using an appropriate workgroup size plays an important role in realizing fast and efficient simulations. Currently, an effective workgroup size is experimentally determined based on the computational capabilities of the CPU versus the GPGPU.

```
begin initialization
    // Generate OpenCL kernel for Heterogeneous Computing
end initialization
begin simulation
    Time LVT = 0, GVT = 0, lastLVT = 0
    std::list<HCAgent*> ocl; // HC agents per time step
    while GVT < endTime do
        LVT = scheduler→getNextEventTime();
        if (LVT > lastLVT) or (ocl.size() > workgroupSize) then
            if kernelRunning then // overlapped operations
                wait() // Wait for previous GPU kernel
                copyStateFromGPU();
            end if
            copyStateToGPU(ocl);
            scheduleOpenCLKernel(); // asynchronous launch
            kernelRunning = true; lastLVT = LVT;
        end if
        // Do discrete event processing and track HC agents
        if processNextEvent() then ocl.push_back(agent);
        GVT = updateGVT(); // Update GVT as needed
    end while
end simulation
```

**Algorithm 1:** Algorithmic overview of salient simulation operations in MUSE-HC

## EXPERIMENTS

The proposed MUSE-HC modeling and simulation framework capable of heterogeneous computing (HC) has been used to conduct a broad range of experiments to assess its effectiveness. The initial set of experiments focused on verification and validation. The performance assessment of MUSE-HC has been conducted using a synthetic benchmark that embodies the key characteristics of temporospatial epidemic models. The design of experiments, observations, and inferences are discussed in the following subsections.

*Hardware platform used for experiments*

The experiments has been conducted on a workstation equipped with Intel Xeon E5-2680 v4 CPU (at 2.4GHz with turbo boost to 3.2 GHz) and a NVIDIA Pascal P100 GPGPU. The GPGPU consists of 3,584 SIMD cores that can deliver a theoretical 4.7 TeraFLOPS of double-precision performance. It has 16 GB of HBM2 memory on the device.

### Verification and Validation (V&V)

The verification and validation (V&V) of MUSE-HC has been conducted using the Ebola model in Figure 7. This model has been extensively validated by Legrand et al. (2007) and more recently by Rivers et al. (2014). Specifically, we have used the Ebola model with epidemic constants for Liberia obtained from Rivers et al. (2014). The inset chart in Figure 8 shows the reference model output from Rivers et al. (2014). The larger chart in the figure shows the corresponding ODE (deterministic) and SSA (stochastic) version of the outputs from MUSE-HC. As illustrated by the charts in Figure 8, the outputs form MUSE-HC were consistent with the expected data. Furthermore, outputs from HC simulations were statistically compared to the results from a purely CPU-based simulation to ensure they produced identical results. Note that statistical comparison of outputs is necessary only for the stochastic version of the simulation due to the inherent randomness in the model's output.

The V&V experiments also included metamorphic testing with different initial epidemic settings. The outputs from HC simulations were statistically compared to the results from a purely CPU-based simulation to ensure both versions produced identical results. Collectively, these experiments verify and validate not only the models produced by EDL parser but also the simulation infrastructure of MUSE-HC.

### Synthetic benchmark for performance assessment

In this study, performance assessments of MUSE-HC have been conducted using a synthetic benchmark. The benchmark has been designed to dynamically generate a grid of agents of specified size. The grid of agent mirrors the typical structure of an temporospatial model, similar to the model shown in Figure 1. Each agent performs deterministic or stochastic simulation using heterogeneous computing (HC) capabilities of MUSE-HC. In addition, each agent exchanges

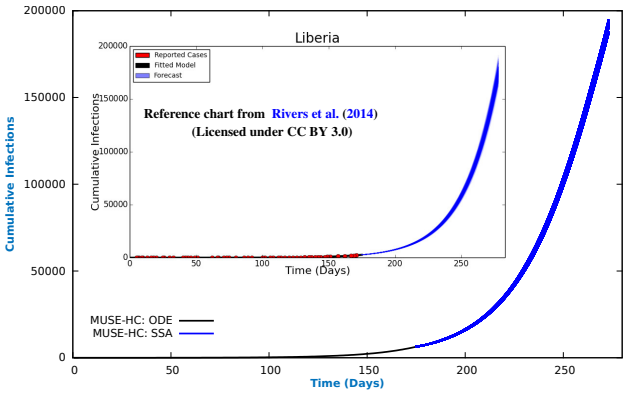



Figure 8: Results from V&V experiments conducted using Ebola model from Rivers et al. (2014) as reference.

a given number of events to simulate movement of people to different geographic regions. The synthetic benchmark provides a comprehensive set of command-line arguments to configure its operational characteristics and eases assessment of different settings. This feature has been used for conducting the performance experiments discussed in the following subsections.

### Influence of OpenCL workgroup size

Workgroup size plays an important role in effective use of heterogeneous computing for two key reasons. First, it influences the hardware utilization on the GPGPU. Small workgroup sizes do not fully utilize the GPGPU hardware while large workgroup sizes increase overheads of transferring data. Second, workgroup sizes influence the degree of overlapped execution between CPU and GPU.

The chart in Figure 9 illustrates the impact of varying the workgroup size on simulation runtime. The chart plots data for both deterministic (using Runge-Kutta) and stochastic (Gillespie with Tau+Leap) simulations. The data plotted is the average from 10 independent runs for each workgroup size. The data has been collected using the synthetic benchmark with 10,000 agents in a 100×100 grid. The data shows that the workgroup size has a more pronounced influence on ODE execution when compared to SSA execution. This is because the ODE version has a much higher computational workload when compared to the corresponding SSA version. However, the improvements tapper off around a workgroup size of 3,000 because the GPGPU has 3,584 SIMD cores and any workgroup larger than that number cannot operate in parallel. However, the larger workgroups are able to better amortize overheads of data transfers between CPU and GPGPU, thereby slightly improving performance. Consequently, for rest of the performance experiments we have used a workgroup size that is equal to the number of agents in the model.

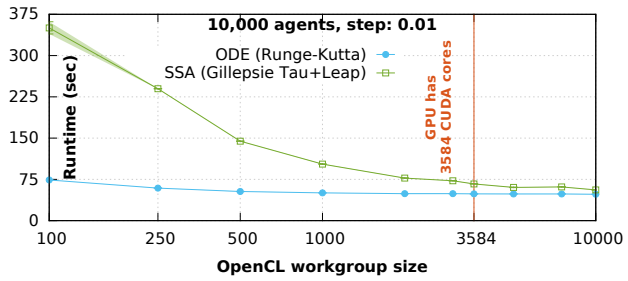
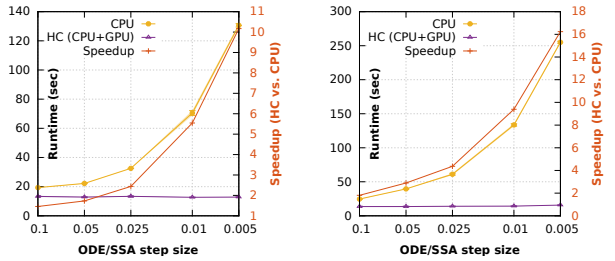



Figure 9: Impact of varying workgroup size

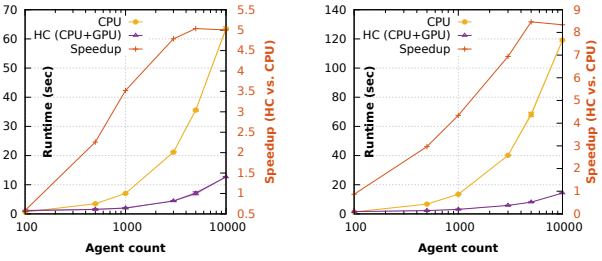



(a) ODE (Runge-Kutta) (b) SSA (Gillespie Tau+Leap)

Figure 11: Impact of varying step size (10,000 agents)

### Effect of model size (*i.e.,* number of agents)

The computational requirements of a simulation is directly proportional to the model's size, *i.e.,* number of agents in a model. The charts in Figure 10 illustrates the influence of model size on the runtime. The charts also compare the runtime of the heterogeneous computing (CPU+GPGPU) versus simulation run using 1 core on the CPU. As illustrated by the charts, for small models with just 100 agents, the HC version does not yield significant performance improvement. This is because the compute capabilities of the GPGPU are not fully utilized. With increase in model size, more agents are scheduled to run on the GPGPU thereby better utilizing its compute resources and the HC version outperforms its corresponding CPU-only version. The maximum speedup for deterministic (*i.e.,* ODE-based) version is 5× while the stochastic (*i.e.,* SSA version) is 8.5×. The SSA version yields better performance improvement due to its increased computational needs arising from random number generation. In both cases, however, the performance peaks around 4000 agents – the peak is attributed to the maximum of 3,584 SIMD cores available on the GPGPU used in this study.




(a) ODE (Runge-Kutta) (b) SSA (Gillespie Tau+Leap)

Figure 10: Impact of varying number of agents (step: 0.01)

### Effect of step size

The computational demands of both deterministic and stochastic simulation are inversely proportional to the step size – *i.e.,* as step size decreases the computational demands increases. Increase in computational needs in turn increases runtime of simulations. The charts in Figure 11 illustrate the change in runtime as the step size is decreased. As illustrated by the CPU-only curves, for both ODE and SSA versions, the runtime increases polynomially as expected. Nevertheless, the increased computational demands at each simulation time enables more effective use of GPGPU resources and the speed-up realized by the heterogeneous computing (HC) version proportionally increased. With a time step of 0.005, the HC version provides 10× and 16× performance improvement for the ODE and SSA versions respectively, highlighting the overall effectiveness of MUSE-HC.

## CONCLUSIONS & FUTURE WORK

Humanity will continue to face multifaceted health and socioeconomic challenges due to communicable and zoonotic diseases. Hence, there is a heightened urgency to develop comprehensive methods for proactively containing epidemics. Contemporary approaches to epidemic forecasting and containment rely on temporospatial modeling and simulation. Temporospatial epidemic models are used to characterize the diversity in ecology, geography, weather, population density, etc. Unfortunately, the accuracy and fidelity of such models is realized at the cost of substantial increase in computational time, which hinders comprehensive analyses in short time frames.

Heterogeneous Computing (HC) holds considerable promise to accelerate simulation of temporospatial models. However, HC currently requires considerable technical skill and development of software for different programming paradigms for both deterministic and stochastic simulations. These issues hinder effective use of HC, particularly by the end-users, including epidemiologists and public health experts.

Accordingly, to ease the effective use of heterogeneous computing (HC) for epidemic simulations, this paper presents a novel modeling and simulation environment called MUSE-HC. MUSE-HC provides a domain-specific modeling language called Epidemic Description Language (EDL) to streamline modeling. The syntax and semantics of EDL have been designed to provide direct mapping to conceptual compartmental models. The paper illustrated the intuitive and expressive constructs supported by EDL using an Ebola model.

A single EDL description is compiled and transformed to generate both ODE and SSA simulations capable of utilizing heterogeneous compute platforms. Automatic code generation eliminates overheads of manual programming. The

generated model utilizes MUSE-HC to perform discrete event processing on a CPU and epidemic equation processing on a GPGPU. The MUSE-HC kernel ensures efficient use of both the CPU and GPGPU by overlapping execution of different agents.

The modeling and simulation infrastructure of MUSE-HC has been verified and validated using a well-established Ebola model. The performance assessment of MUSE-HC has been conducted using a synthetic model that eases exploration of different model settings. A key hardware configuration that influences the utilization of GPGPU is the workgroup size. The experiments showed that large workgroups sizes are better because it enables faster amortization of the overheads involved in copying data between the CPU's memory and the GPGPU's memory. The number of agents in the model also had strong influence on the performance gains realized using heterogeneous computing (HC). However, the step size used by numerical methods had the most conspicuous impact on performance improvements realized using HC. With a time step of 0.005, the HC version provides 10× and 16× performance improvement for the ODE and SSA versions respectively, highlighting the overall effectiveness of MUSE-HC.

The experiments show that additional opportunities exist to improve performance of heterogeneous computing by further overlapping CPU and GPGPU computation. We plan on exploring this aspect in future works. Importantly, we also plan to pursue parallel simulations using a cluster of workstations capable of heterogeneous computing. Parallel, heterogeneous computing approaches also holds considerable promise to further accelerate simulation of large temporospatial epidemic models.

## ACKNOWLEDGMENTS

Thanks to Harrison Roth (rothhl@miamiOH.edu) for initial prototype and assessment of MUSE-HC.

Support for this work was provided in part by the Ohio Supercomputer Center (Grant: PMIU0110-2).

## REFERENCES

Arlindo G.F.; dePaula Lauro M.; Jos C.C.; Woerle L.T.; and Anderson S.S., 2015. *CUDA parallel programming for simulation of epidemiological models based on individuals*. *Mathematical Methods in the Applied Sciences*, 39, no. 3, 405–411. doi:10.1002/mma.3490.

Giridharan N. and Rao D.M., 2016. *Eliciting Characteristics of H5N1 in High-Risk Regions Using Phylogeography and Phylodynamic Simulations*. *Computing in Science and Engineering*, 18, no. 4, 11–24. ISSN 1521-9615. doi:10.1109/MCSE.2016.77.

Legrand J.; Grais R.F.; Boelle P.Y.; Valleron A.J.; and Flahault A., 2007. *Understanding the dynamics of Ebola epidemics*. *Epidemiology and Infection*, 135, no. 4, 610–621. doi:10.1017/S0950268806007217.

Leonenko V.N.; Pertsev N.V.; and Artzrouni M., 2015. *Using High Performance Algorithms for the Hybrid Simulation of Disease Dynamics on CPU and GPU*. *Procedia Computer Science*, 51, 150 – 159. ISSN 1877-0509. doi:https://doi.org/10.1016/j.procs.2015.05.214. URL http://www.sciencedirect.com/science/article/pii/S1877050915010224. International Conference On Computational Science, ICCS 2015.

Rao D.M., 2016. *Efficient parallel simulation of spatially-explicit agent-based epidemiological models*. *Journal of Parallel and Distributed Computing*, 93-94, 102–119. ISSN 0743-7315. doi:10.1016/j.jpdc.2016.04.004.

Rao D.M., 2018. *Performance Comparison of Cross Memory Attach Capable MPI vs. Multithreaded Optimistic Parallel Simulations*. In *Proceedings of the 2018 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. ACM, New York, NY, USA, SIGSIM-PADS '18. ISBN 978-1-4503-5092-1, 37–48. doi:10.1145/3200921.3200935. URL http://doi.acm.org/10.1145/3200921.3200935.

Rao D.M.; Asbun C.; and Stamper P.D., 2017. *Forecasting and Assessment of Autochthonous Yellow Fever Outbreak in Brazil*. In *66th Annual Meeting*. American Society for Tropical Medicine and Hygiene, Baltimore Convention Center, Baltimore, Maryland, USA. doi:10.13140/RG.2.2.18785.40805. Poster.

Rao D.M. and Chernyakhovsky A., 2013. *Automatic Generation of Global Agent-based Model of Migratory Waterfowl for Epidemiological Analysis*. In *Proceedings of the 27th European Simulation and Modelling Conference (ESM'2013)*. EuroSis, Lancaster University, Lancaster, UK. Best paper award.

Rivers C.M.; Lofgren E.T.; Marathe M.; Eubank S.; and Lewis B.L., 2014. *Modeling the Impact of Interventions on an Epidemic of Ebola in Sierra Leone and Liberia*. *PLOS Current Outbreaks*. doi:10.1371/currents.outbreaks.fd38dd85078565450b0be3fcd78f5ccf.

Shekh B.; Doncker E.D.; and Prieto D., 2015. *Hybrid multithreaded simulation of agent-based pandemic modeling using multiple GPUs*. In *2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. 1478–1485. doi:10.1109/BIBM.2015.7359894.

Yeom J.S.; Bhatele A.; Bisset K.; Bohm E.; Gupta A.; Kale L.V.; Marathe M.; Nikolopoulos D.S.; Schulz M.; and Wesolowski L., 2014. *Overcoming the Scalability Challenges of Epidemic Simulations on Blue Waters*. In *Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium*. IEEE Computer Society, Washington, DC, USA, IPDPS '14. ISBN 978-1-4799-3800-1, 755–764.

Zou P.; shuai L Y.; da Wu L.; li Chen L.; and ping Yao Y., 2013. *Epidemic simulation of a large-scale social contact network on GPU clusters*. *SIMULATION*, 89, no. 10, 1154–1172. doi:10.1177/0037549713482026.