MoleCom-gpu: A GPU-based Parallel Simulation Framework for Molecular Communications

Dept. of Computer Engineering
Computer Networks Laboratory
(NETLAB)

Bogazici University

Prof. Tuna Tugcu
tugcu@boun.edu.tr

May 11, 2017

Content

Introduction

- A customizable simulation tool for molecular communication
- Utilizes GPU for parallel processing
- An open-source project for the use and contribution of all nanonetworking society
- Encourage the nanonetworking research community to freely use and improve the simulator

Introduction
Motivation

- Re-designing the simulator for each paper/project is waste of researchers' valuable time
- Reproducibility of the results required for comparison of alternative approaches
- Need for a shared platform that will serve as a benchmark environment for fair and meaningful testing
- Large number of molecules, all exhibiting similar movement behaviour, hints parallel processing

Why GPU programming

- GPU-accelerated computing is the use of a graphics processing unit (GPU) together with a CPU to accelerate deep learning, analytics, and engineering applications[1]
- GPUs have highly parallel architecture, accompanied by thousands of processing cores
- On the other hand, CPUs utilize a few cores for (almost) sequential serial processing of data

[1] **http://www.nvidia.com/object/what-is-gpu-computing.html**.

Why GPU programming

- The structure of molecular communication simulation consists of huge number of molecules moving around
- Every molecule is doing computationally simple action, and these actions are repeated for a very large number of molecules in the environment
- This nature of the problem makes it suitable for parallel GPU computation
- x10s average performance gain, compared to traditional CPU programs

Basic Design
Workflow

- Generate simulation environment
- Run and collect simulation data with generated environment

Basic design
Simulation environment

- Generated separately via configGenerate.lua script
- The parameters for the simulation and properties of the simulation entities are provided here
- configGenerate.lua outputs the environment as a separate file
- This file is then used as the input for main simulator script

Basic design
Simulation environment variables

The following variables are configurable in environment generation

- Numbers of the transmitters and the receivers
- Coordinates of the transmitters and the receivers
- Radii of the transmitters and receivers
- Radius of carrier molecules
- Diffusion coefficient

Basic Design
Simulation Variables

The following simulation related variables are configurable in environment generation

- Simulation step size
- Simulation duration
- Symbol size
- Symbol duration

Basic Design
Simulation Output

Simulation is run via generated environment file

- Time vs. number of received molecules
- User can fit his/her own modulation scheme to raw results

Setup
Configuration

- CPU : Intel Core i5 6500
- GPU : NVidia GeForce GTX 1080
- RAM : Kingston HyperX 8 GB 2133 MHz
- Storage : Samsung 850 EVO SSD 256GB
- OS : Ubuntu 16.04 LTS

Total cost : 1500 $

Setup
Software

- NVidia CUDA
- torch7 - Scientific computation library in LUA scripting language
- CUtorch 1.0.0 - CUDA backend for torch7

- We verify our simulation model with the analytical model for a point transmitter and a spherical receiver described in Yilmaz *et al.*[2]
- We compare the speed-up of our simulator with MUCIN[3] , a molecular communication simulator written in MATLAB and runs on CPU

[2] **H. Birkan Yilmaz**; **Akif Cem Heren**; **T. Tugcu**; **and Chan-Byoung Chae**; **"Three-Dimensional Channel Characteristics for Molecular Communications With an Absorbing Receiver" IEEE Communications Letters**; **vol. 18**; **iss. 6**; **pp. 929 - 932**; **2014.**.

[3] **https://www.mathworks.com/matlabcentral/fileexchange/46066-molecular-communication–mucin–simulator**.

Verification

- We run the simulation with two different transfer models with six different scenarios
- Scenarios have different:
  - Receiver radius (R_r)
  - Distance (r_0)
  - Diffusion coefficients (D)

Verification
Transfer Models

- Point transmitter - Spherical receiver
- This transmission is analytically modeled in the paper[4]
- Point transmission on a spherical transmitter - Spherical receiver
- This model is compared with the analytical case without spherical transmitter
- As will be seen in the results, if the transmission point is between transmitter and receiver, the number of received molecules are higher than the analytical model
- Otherwise, the number of received molecules are lower than the analytical model

[4] **H. Birkan Yilmaz**; **Akif Cem Heren**; **Tuna Tugcu**; **and Chan-Byoung Chae**; **"Three-Dimensional Channel Characteristics for Molecular Communications With an\leC** **Vol 18**; **No 6**; **pp. 929 - 932**; **2014.**.

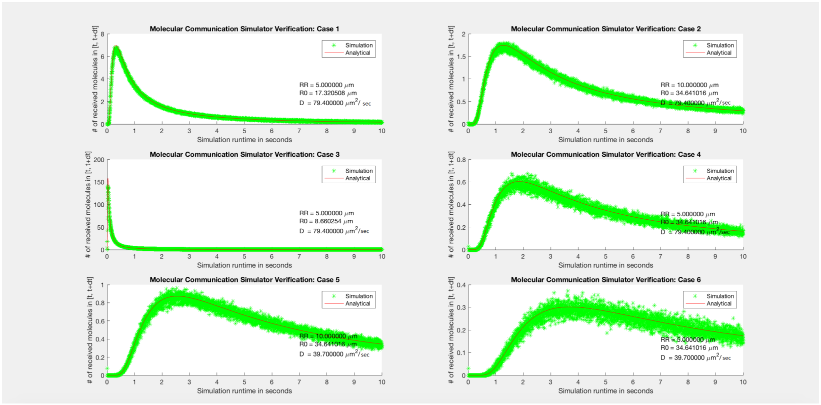
Verification

Scenarios

Scenario	R_r	r_0	D
Scenario 1	5.0 µm	17.32 µm	79.40 $\mu m^2/sec$
Scenario 2	10.0 µm	34.64 µm	79.40 $\mu m^2/sec$
Scenario 3	5.00 µm	8.66 µm	79.40 $\mu m^2/sec$
Scenario 4	5.00 µm	34.64 µm	79.40 $\mu m^2/sec$
Scenario 5	10.00 µm	34.64 µm	39.70 $\mu m^2/sec$
Scenario 6	5.00 µm	34.64 µm	39.70 $\mu m^2/sec$

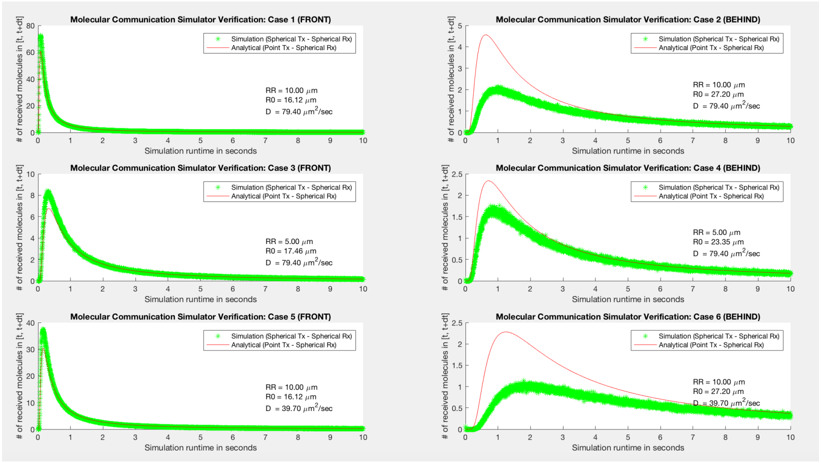
Results

Verification (Spherical Receiver - Point Transmitter)



Results

Verification (Spherical Receiver - Point Transmission on a Spherical Transmitter)

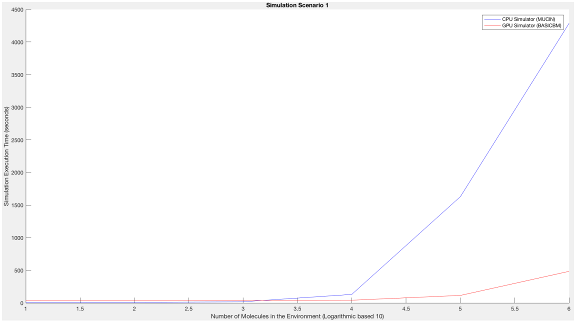


Results
Speedup

- The speed of simulation depends on
  1. Simulation duration
  2. Step time
  3. Number of molecules in the environment
- Diffusion coefficient, distance between transmitters, and molecule size do not significantly affect the execution time
- The dependence on simulation duration and step time is linear
- What about the number of molecules?

Results
Speedup - Scenario 1

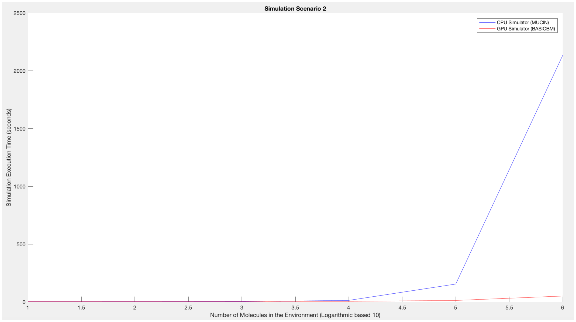
- $\Delta t = 10^{-5}$ seconds, $t_s = 1$ second
- Hence, *numberOfOperations* $= 10^5$



Results

Speedup - Scenario 2

- $\Delta t = 10^{-4}$ seconds, $t_s = 1$ second
- Hence, $\mathit{numberOfOperations} = 10^4$



Results
Speedup

- Observe that the execution time linearly depends on the number of operations
- The number of molecules in the environment exponentially increases the execution time of the CPU Simulator
- As we utilize parallel processing in our simulator, its effect is also nearly linear for us
- The speed-up factor depends on the number of messenger molecules, ranging from 1 to 200 in Scenario 2

Future Work

- Different molecule types for receiving
- Absorbing/Reflecting boundaries
- Multiple transmitters/receivers

Github

- `http://github.com/MoleCom-gpu/MoleCom-gpu`
- Communication through `mailto:MoleCom-gpu@listeci.cmpe.boun.edu.tr`

Thanks for listening

You are all welcome to use and contribute