Simulating the Spread of Epidemics in Real-world Trading Networks using OpenCL

Martin Clauß - martin@dev2day.de

December 2, 2011
Simulating Epidemics with OpenCL

Martin Clauß - martin@dev2day.de

Outline

1 Introduction

2 Related Work

3 Mathematical Model

4 Data and Data Structures

5 OpenCL

6 Simulation

7 Results

8 Future Work
Introduction
Problem

Disease

1. Animal Transport
2. Farm

Simulating Epidemics with OpenCL
Related Work
NetEvo2 Implementation

- NetEvo2-based simulator from Peter A. Kolski (Friedrich-Loeffler-Institut / Bundesforschungsinstitut für Tiergesundheit)
- NetEvo2 is a software library to simulate complex dynamical networks (written by Thomas Gorochowski, Bristol Centre for Complexity Sciences)
- NetEvo2 uses **odeint v2** to solve ordinary differential equations (written by Karsten Ahnert and Mario Mulansky, University of Potsdam, Department of Physics and Astronomy)
- The program is used for performance comparison
Mathematical Model
SIR Model

susceptible → infectious → recovered
SIR Model

\[
\frac{dS}{dt} = -\beta IS \\
\frac{dI}{dt} = \beta IS - \nu I \\
\frac{dR}{dt} = \nu I
\]
SIR Model

\[ \frac{dS}{dt} = -\beta IS \]
\[ \frac{dI}{dt} = \beta IS - \nu I \]
\[ \frac{dR}{dt} = \nu I \]

where

- \( \beta \) is the contact rate and
- \( \nu \) is the recovery rate
Data and Data Structures
## Data and Data Structures

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Amount</th>
<th>Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>10002</td>
<td>12</td>
<td>1001</td>
</tr>
<tr>
<td>10033</td>
<td>12233</td>
<td>4</td>
<td>1001</td>
</tr>
<tr>
<td>10231</td>
<td>10333</td>
<td>5</td>
<td>1002</td>
</tr>
<tr>
<td>10002</td>
<td>10000</td>
<td>22</td>
<td>1004</td>
</tr>
<tr>
<td>From</td>
<td>To</td>
<td>Amount</td>
<td>Day</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>--------</td>
<td>-----</td>
</tr>
<tr>
<td>10000</td>
<td>10002</td>
<td>12</td>
<td>1001</td>
</tr>
<tr>
<td>10033</td>
<td>12233</td>
<td>4</td>
<td>1001</td>
</tr>
<tr>
<td>10231</td>
<td>10333</td>
<td>5</td>
<td>1002</td>
</tr>
<tr>
<td>10002</td>
<td>10000</td>
<td>22</td>
<td>1004</td>
</tr>
<tr>
<td>From</td>
<td>To</td>
<td>Amount</td>
<td>Day</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>--------</td>
<td>-----</td>
</tr>
<tr>
<td>10000</td>
<td>10002</td>
<td>12</td>
<td>1001</td>
</tr>
<tr>
<td>10033</td>
<td>12233</td>
<td>4</td>
<td>1001</td>
</tr>
<tr>
<td>10231</td>
<td>10333</td>
<td>5</td>
<td>1002</td>
</tr>
<tr>
<td>10002</td>
<td>10000</td>
<td>22</td>
<td>1004</td>
</tr>
<tr>
<td>From</td>
<td>To</td>
<td>Amount</td>
<td>Day</td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>10000</td>
<td>10002</td>
<td>12</td>
<td>1001</td>
</tr>
<tr>
<td>10033</td>
<td>12233</td>
<td>4</td>
<td>1001</td>
</tr>
<tr>
<td>10231</td>
<td>10333</td>
<td>5</td>
<td>1002</td>
</tr>
<tr>
<td>10002</td>
<td>10000</td>
<td>22</td>
<td>1004</td>
</tr>
</tbody>
</table>
## Data and Data Structures

### Simulating Epidemics with OpenCL

#### Table: Arcs IDs

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Amount</th>
<th>Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>10002</td>
<td>12</td>
<td>1001</td>
</tr>
<tr>
<td>10033</td>
<td>12233</td>
<td>4</td>
<td>1001</td>
</tr>
<tr>
<td>10231</td>
<td>10333</td>
<td>5</td>
<td>1002</td>
</tr>
<tr>
<td>10002</td>
<td>10000</td>
<td>22</td>
<td>1004</td>
</tr>
</tbody>
</table>

#### Day and Active Arcs (IDs)

<table>
<thead>
<tr>
<th>Day</th>
<th>Active Arcs (IDs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1002</td>
<td>2</td>
</tr>
<tr>
<td>1004</td>
<td>3</td>
</tr>
</tbody>
</table>

![Diagram showing the simulation of epidemics with OpenCL]
Simulating Epidemics with OpenCL
indexes / node IDs

_global nodes

0 1 2 3 4 5

indexes

_global incomingNodes

1 2 3 4 5

indexes

_global incomingArcs

0 1 2 3 4

Martin Clauß - martin@dev2day.de

Simulating Epidemics with OpenCL
Data Structures for OpenCL

indexes / node IDs
__global nodes
0 3 3 3 4 5

indexes
__global incomingNodes
1 2 3 4 5

indexes
__global incomingArcs
0 1 2 3 4 5

Simulating Epidemics with OpenCL
Data Structures for OpenCL

indexes / node IDs
__global nodes
0 1 2 3 4 5

indexes
__global incomingNodes
0 1 2 3 4 5

indexes
__global incomingArcs
0 1 2 3 4

Simulating Epidemics with OpenCL
Data Structures for OpenCL

indexes / node IDs

_global nodes

indexes

__global incomingNodes

indexes

__global incomingArcs

indexes

__global arcActivity

indexes

0 1 2 3 4 5

0 3 3 3 4 5

1 2 3 4 5

1 2 3 4 5

0 1 2 3 4 5

0 1 2 3 4 5

0 1 2 3 4 5

0 1 2 3 4 5

per day

Simulating Epidemics with OpenCL

Martin Clauß - martin@dev2day.de
Simulating Epidemics with OpenCL
OpenCL

- **Open Computing Language**
- Standard for **heterogeneous parallel** computing developed by the Khronos Group
- Cross-platform and cross-vendor development
- Support for a wide variety of devices (multicore CPUs, GPUs, DSPs, Cell/B.E.)
- The programming language **OpenCL C** is a subset of ANSI-C99 with a couple of additions and restrictions
OpenCL Devices

Host

Compute Device

Compute Unit

Processing Element

Martin Clauß - martin@dev2day.de

Simulating Epidemics with OpenCL
OpenCL Memory Model

The diagram illustrates the memory model in OpenCL, with the following components:

- **Host Memory**
- **Global / Constant Memory**
- **Local Memory**
- **Private Memory**
  - **Work Item**
  - **Work Group**

**Compute Device** is connected to **Host Memory** via **PCIe (@ 4-5 GB/s)**.
OpenCL Index Range

Image Source: The OpenCL Specification Version 1.1
Serial C vs. OpenCL C

__kernel void square(__global int *input, __global int *output)
{
    int i  = get_global_id(0);
    output[i] = input[i] * input[i];
}
for (unsigned int i = 0; i < size; i++)
{
    output[i] = input[i] * input[i];
}

for (unsigned int i = 0; i < size; i++)
{
    output[i] = input[i] * input[i];
}
Simulation
numberOfDaysToSimulate ← 10

currentDay ← 0

while currentDay < numberOfDaysToSimulate or not termination criterion is met do
  simulateLocal() {compute ODEs per node}
  simulateInteraction() {compute trading transactions between nodes}
  currentDay ← currentDay + 1

end while
\begin{verbatim}
numberOfDaysToSimulate ← 10
currentDay ← 0
while currentDay < numberOfDaysToSimulate or not termination criterion is met do
    simulateLocal() \{compute ODEs per node\}
    simulateInteraction() \{compute trading transactions between nodes\}
    currentDay ← currentDay + 1
end while

numberOfIntegrationSteps ← 1000
stepSize ← 0.001
t ← 0 \{time\}
for \( i = 0 \) to numberOfIntegrationSteps do
    executeRungeKutta4Kernel(stepSize, t) \{Solves the ODEs of all nodes using Runge-Kutta 4th order\}
    t ← t + stepSize
end for
\end{verbatim}
numberOfDaysToSimulate ← 10

currentDay ← 0

while currentDay < numberOfDaysToSimulate or not termination criterion is met do
    simulateLocal() {compute ODEs per node}
    simulateInteraction() {compute trading transactions between nodes}
    currentDay ← currentDay + 1
end while

Build the binary array of arc activity for currentDay
(e.g. 1, 0, 0, 0, 1, 0, ..., 0 means arcs 0 and 4 are active)
executeInteractionKernel {The interaction kernel gets called only once}
numberOfDaysToSimulate ← 10

currentDay ← 0

while currentDay < numberOfDaysToSimulate or not termination criterion is met do
    simulateLocal() {compute ODEs per node}
    simulateInteraction() {compute trading transactions between nodes}
    currentDay ← currentDay + 1
end while

Build the binary array of arc activity for currentDay
(e.g. 1, 0, 0, 0, 1, 0, ..., 0 means arcs 0 and 4 are active)
executeInteractionKernel {The interaction kernel gets called only once}

{Every node is handled by a single thread}
nodeID ← threadID

for all incoming nodes of node nodeID do
    if the arc (incomingNode, nodeID) is active on currentDay then
        transport animals and update S, I, R of node nodeID
    end if
end for
Simulation

Simulating Epidemics with OpenCL

Martin Clauß - martin@dev2day.de
Simulation

Simulating Epidemics with OpenCL

Discrete time step

Current node

Data access: write

Incoming animals

Data access: read

S, I, R

S, I, R

Martin Clauß - martin@dev2day.de
Results
Results

- All programs have been compiled with `g++ 4.5` and an optimization level of `-O3`.
- **CPU**: Intel Core i7 950 @ 3.07GHz
- **GPU**: GeForce GTX 470
- **Serial** program: NetEvo2 implementation
- **Parallel** program: EpidemicOCL
- **Double Precision vs. Single Precision**: Error < $3 \times 10^{-5}$
Results

Performance Comparison

- **EpidemicOCL on GTX 470 Double Precision**
- **NetEvo2 on Intel Core i7 950 @ 3.07GHz**
- **EpidemicOCL on GTX 470 Single Precision**

Performance Comparison (Logarithmic Scale)

- **EpidemicOCL on GTX 470 Double Precision**
- **NetEvo2 on Intel Core i7 950 @ 3.07GHz**
- **EpidemicOCL on GTX 470 Single Precision**
Results

Simulating Epidemics with OpenCL

Martin Clauß - martin@dev2day.de
Results

- What do those performance measures mean?

Divide the day in 100 time slices or 100 numerical Runge-Kutta steps, respectively. 100 numerical steps are equivalent to a resolution of approximately 15 minutes (24 × 60 / 100 = 14.4). 1,000,000 nodes:

- 1 day: ≈ 136ms
- 1 year: ≈ 48s

Realistic example: 100,000 nodes (e.g. pigs)

- 1 day: ≈ 14ms
- 1 year: ≈ 6s
What do those performance measures mean?

- Divide the day in 100 time slices or 100 numerical Runge-Kutta steps, respectively
- 100 numerical steps are equivalent to a resolution of approximately 15 minutes ($24 \times 60/100 \approx 14.4$)
- 1,000,000 nodes
- 1 day: $\approx 136$ms
- 1 year: $\approx 48$s
What do those performance measures mean?

- Divide the day in 100 time slices or 100 numerical Runge-Kutta steps, respectively
- 100 numerical steps are equivalent to a resolution of approximately 15 minutes \(24 \times \frac{60}{100} = 14.4\)
- 1,000,000 nodes
- 1 day: \(\approx 136\)ms
- 1 year: \(\approx 48s\)

- Realistic example: 100,000 nodes (e.g. pigs)
- 100 numerical steps as the resolution for 1 day
- 1 day: \(\approx 14\)ms
- 1 year: \(\approx 6s\)
Future Work
Future Work

- Visualization
- Interaction - modification of model parameters in real-time
- Redesign / Open Source Release
- Video
Thank You!