

Centre for Water Systems, College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, UK
Project Partners & Team

Dr Albert Chen  Dr Bidur Ghimire  Dr Michele Guidolin  Miss Rebecca Austin  Mr Mike Gibson  Mr Andrew Duncan

Prof Dragan Savic  Prof Slobodan Djordjevic  Dr Edward Keedwell

City of Bradford MDC  Environment Agency  Halcrow  HR Wallingford  Mouchelii  MWH  UKWIR  United Utilities

Northumbrian Water  Raa  South West Water  Thames Water  Torbay Council  MWH Soft  Yorkshire Water

University of Exeter  Centre for Water Systems
Outline

- Conventional pluvial flood modelling
- CADDIES project
  - Cellular Automata Dual Drainage Simulation
  - Cellular Automata (CA) modelling alternative
- CA Framework (API)
  - Research tool to allow CA models to be developed
  - Parallelisation (GPU / Cluster / Cloud)
  - CA model performance evaluation
Conventional 1D/2D Pluvial Flood Modelling

- Hydrodynamic (St Venant equations)
  - Solution is iterative process
  - Urban surface / sewer flow
    - “dual drainage”
  - Full 2D approach
    - computationally intensive (hours run time)
- Pseudo-2D
  - 1D surface channels + ponds
- Trade-off between speed and accuracy
CADDIES project
Cellular Automata Dual Drainage Simulation

- Improve “Dual-Drainage” flood modelling
  - Urban surface / sewer flow

- Cellular Automata
  - Simulate complex physical systems using simple rules
  - Fast computations
Cellular automata

- Grid of cells
- Each cell has a set of states
- Simple transition rules on each cell
- Use previous state of neighbourhood
- Ideal for parallelisation
- Famous example
  - Game of life (glider gun)
CADDIES Challenges

To simulate complex flooding processes using simple rules... not trivial!

- Grid of cells – each with a “state”
  - e.g. flood depth / flow rate / water volume
- Cell states evolve stepwise
  - simultaneously across whole grid
- New state of each cell
  - only determined by states of immediate neighbours in previous timestep
CADDIES Challenges

- CA Algorithm is defined by:
  - Transition rules (based on states of neighbours)
    - How many states and which types of value?
    - Which relationships / transition functions?
    - How to handle grid boundary cells?
  - Underlying grid structure
    - Which types of grid structure? (regular / irregular)
    - Cell shapes? (polygon sides)
    - How many cells in the neighbourhood?
CADDIES Framework

- A new flexible software environment (various tools)
  - API (Application Programming Interface)
  - that aims to:
    - Simplify the development/test/analysis of CA algorithms
    - Accelerate CA algorithm using modern hardware
    - Graphically manage the development of CA algorithms
CADDIES CA API

- Standard set of
  - Data structures
  - Variables
  - Methods
CADDIES CA API Ideas

• The developer need write the CA algorithm only once
• The algorithm will work with different types of grids

• Automatically accelerate execution using modern hardware (GPU / Cluster / Cloud)
CA ALGORITHM using the API

1. CA Execution
   - Data management
   - Flow control

2. CA Transition Rules
   - Computation in each cell
CA ALGORITHM using the API

3. CA options

- Define Attributes
- Grid/Cells/Etc.
CA API Implementations

- Different implementations depending on:
  - CA attributes
  - Hardware used

- At the moment three implementations:
  - Sequential CPU / Square / Moore
  - Sequential CPU / Square / von Neumann
  - OpenCL GPU / Square / von Neumann
CPU versus GPU
Game of Life Example

CA Execution

```c
#include CA_2d_INCLUDE(gol)

int main(...){
    ...
    CA::Grid GRID(ncols,nrows,1);
    CA::CellBuffState A(GRID), B(GRID);
    CA::Box area(GRID.box());
    ...
    while(loop) {
        CA::Execute::function(area,gol,GRID,A,B);
        ...
    }
}
```

CA Transition Rules

```c
CA_FUNCTION gol(CA_GRID grid,
                 CA_CELLBUFF_STATE_IO a,
                 CA_CELLBUFF_STATE_I b)
{
    CA_GRID_INIT(grid);
    CA_ARRAY_CREATE(grid,CA_STATE,states,caNeighbours+1);
    ...
    caReadCellBuffStateCellArray(grid,b,states);
    for(int i=0;i<=caNeighbours;i++) {
        ...
    }
    caWriteCellBuffState(grid,a,0,states[0]);
}
```
Game of Life Results

Computation Time Serial vs GPU

- CADDIES Serial
- CADDIES OpenCL GPU

Number of Cells vs Time in Seconds
CA Flood algorithm Example

- Computes outflow of cell by ranking the water surface elevation

- Presented in detail in HIC 2012 paper:
  - Colleague: Dr. Bidur Ghimire
  - Wednesday 18 July 9.20 - 9.40
  - Topic A5.2
Case Study

- Stockbridge area in Keighley (UK)
- DEM of 2m resolution with 377x269 cells
- Rainfall of 42.3 mm/hr (100 year return)
- Used two CA API implementations
  - Sequential CPU / Square / von Neumann
  - OpenCL GPU / Square / von Neumann
- Used two hardware configurations
- Used physical based UIM model as reference
## CA flood algorithm Results

<table>
<thead>
<tr>
<th>Model</th>
<th>UIM</th>
<th>Simple Serial</th>
<th>OpenCL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (s)</td>
<td>Time (s)</td>
<td>$S_p^{UIM}$</td>
</tr>
<tr>
<td>1</td>
<td>10371.1</td>
<td>280.9</td>
<td>36.9</td>
</tr>
<tr>
<td>2</td>
<td>5885.7</td>
<td>161.6</td>
<td>36.4</td>
</tr>
</tbody>
</table>
Conclusion

- CA API is part of CADDIES framework
- It possible to simply develop CA algorithms
  - Game of life
  - CA algorithms for pluvial flood modelling
- Automatically accelerate algorithm using modern hardware
- CA flood model with GPU can be an order of magnitude faster than physical models (UIM)
CADDIES is an EPSRC Funded Project (Number: GR/J09796)
Thanks

M.Guidolin@exeter.ac.uk
http://centres.exeter.ac.uk/cws