# Enabling multi-node MPI parallelisation of the LISFLOOD flood inundation model

## ARCHER eCSE12-17

Dr. Arno Proeme (a.proeme@epcc.ed.ac.uk)
EPCC, University of Edinburgh

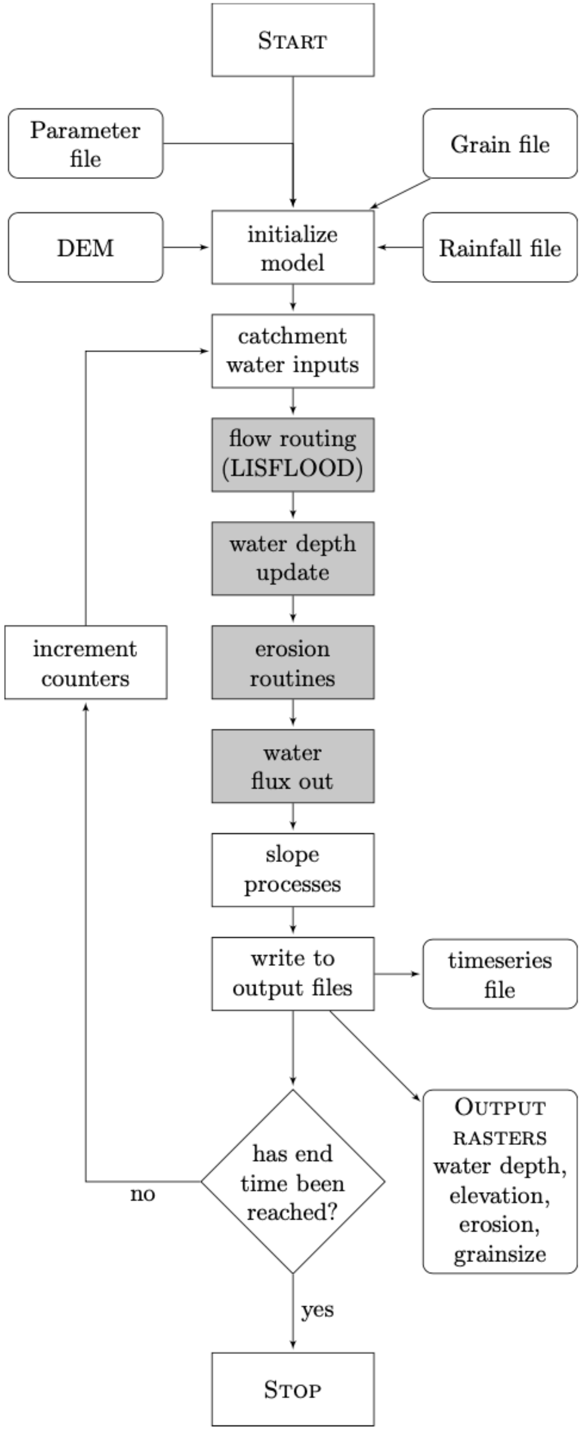
Dr. Declan Valters (British Geological Survey)
Prof. Simon Mudd (Geosciences, University of Edinburgh)

# Landscape evolution modelling & HPC

- Landscape evolution modelling community new to HPC
  - Geomorphology (e.g. erosion, sedimentation, etc.)
  - Hydrology (river flow, flooding, etc.)
- Growing availability of increasingly higher-resolution data
  - Topographic data: e.g. LiDAR surface elevation maps, sub-metre resolution
  - Weather/climate data: sensor data (e.g. rainfall) or simulation outputs
- Great potential from using HPC and high-resolution data:
  - More spatially & temporally detailed processes
  - Higher accuracy
  - Larger domains
  - Shorter time to solution (critical for impact of short-term forecasts)
- Most numerical landscape evolution modelling software not ready to use HPC (limited parallelisation)

# (CAESAR-)LISFLOOD

- Hydrodynamic model
  - Simulates flooding in river catchments and floodplains, erosion & sediment transport processes (optional)
  - Can simulate timescales of hours to 100s of years (geomorphology)
- Enables flood inundation modelling & flood risk research
  - NERC strategic research area
- Previously implemented in **HAIL-CAESAR** by D. Valters (http://dvalts.io/HAIL-CAESAR/)
  - OpenMP-parallelised – limited to single node
- Want to enable multi-node parallelisation of HAIL-CAESAR




# HAIL-CAESAR

From: Valters, D. A (2017). *Modelling catchment sensitivity to rainfall resolution and erosional parameterisation in simulations of flash floods in the UK.* PhD Thesis, University of Manchester.

Simplified outline of HAIL-CAESAR program flow:

- Grey shaded boxes = OpenMP-parallelised code
- Rounded rectangles = input & output files

DEM = Digital Elevation Model (surface elevation, i.e. topography data)

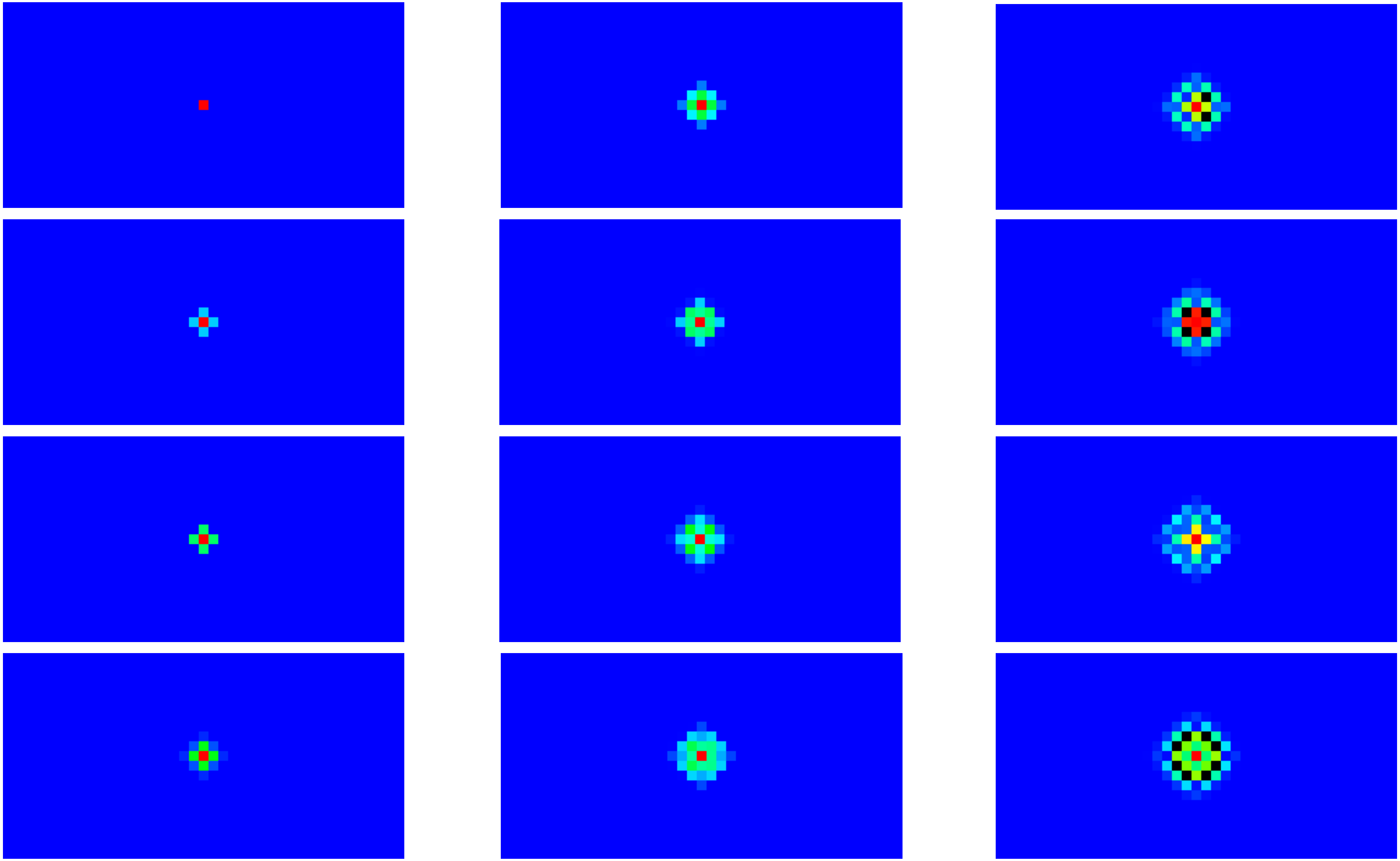
- Focus on multi-node parallelisation of hydrology, i.e.
  - **flow routing (LISFLOOD)**
  - **water depth update**
  - (water flux out)
- Erosion routines secondary
  - no real additional complexity

# HAIL-CAESAR

- 2D cellular automaton / stencil code:
  - Elevation, water depth, other real-valued physical quantities (e.g. fluxes) defined for each cell on a 2D grid
  - Fixed update rule: new value of each main cell quantity depends only on old value and four-point neighbour values (East, West, North, South)
  - Solves a simplified version of the Saint-Venant shallow water equations for 2D depth-averaged flow, calculating water discharge based on local gradients of water depth and bed elevation from neighbouring grid cells
    - takes into account e.g. surface & subsurface discharge, “soil moisture store”, etc.

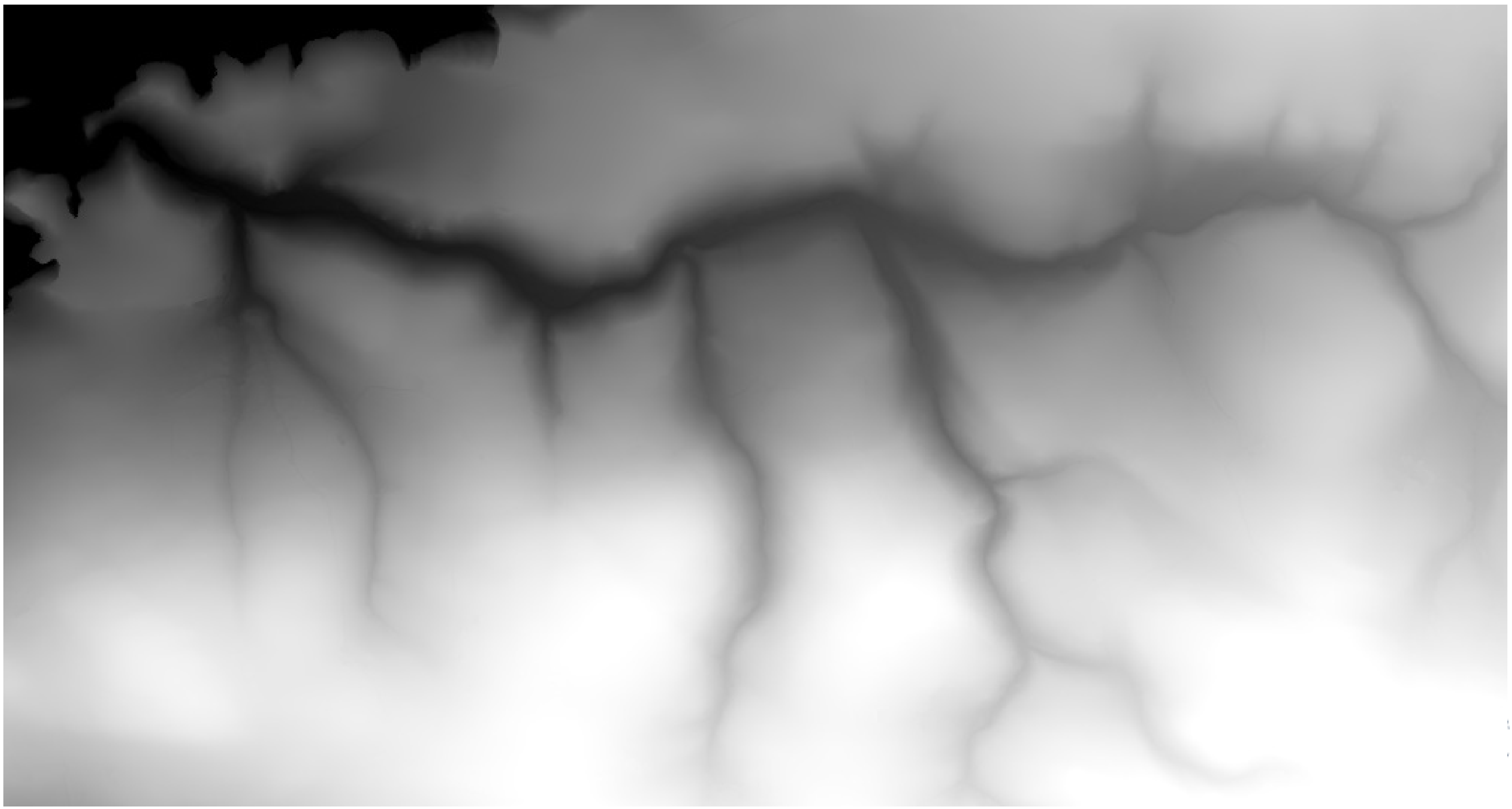
# HAIL-CAESAR hydrology

## water depth evolution for synthetic test case: persistent rainfall on central cell with flow routing



# Realistic Digital Terrain

Boscastle River Valency (Cornwall): $12km^2$, $1m^2$ resolution



# Multi-node parallelisation of LISFLOOD

- Regular grid stencil codes very well defined as a class of problems, and parallelisation approaches well established
  - domain decomposition + halo exchange
- Should be able to leverage existing solution instead of creating $(n+1)^{th}$ reimplementation
  - Library, DSL, …
- Considerations / requirements:
  - Should be based on MPI parallelism
  - Should incorporate dynamic load balancing (load distribution is initially predictable but can change drastically due to flooding or gradually over geomorphological timescales)

# LibGeoDecomp (http://libgeodecomp.org/)

- C++ framework for parallelisation mainly of stencil codes
  - Pure C++, not a DSL, customisable/extendable for Multiphysics
  - Uses Boost library

- MPI based, alternatively also supports:
  - OpenMP (single shared-memory node)
  - *or* CUDA (single GPU)
  - *or HPX* (also developed by Stellar group - http://stellar-group.org/)

- Handles domain decomposition, dynamic load balancing
  - Recursive bisection, Hilbert & zip-zag space-filling curves, Scotch graph-based partitioning, …

# LibGeoDecomp (http://libgeodecomp.org/)

- Optimisations:
  - Overlap computation & communication (latency hiding)
  - Fast iteration through Arrays of Structs (actually stored as SoAs) using instruction set-specific vectorization templates in LibFlatArray (http://www.libgeodecomp.org/libflatarray.html)
- Tested on a number of large HPC systems, possible to obtain good efficiency on (tens of) thousands of cores
- MPI IO-based checkpointing functionality
- Some parallel IO including for visualisation
  - VisIt BOV & Silo formats

# HAIL-CAESAR original

- Read in elevation data from DEM file
- Store elevation and water depth grids in 2D double arrays
- LISFLOOD algorithm loops over arrays
  (OpenMP-parallel if enabled)
- Done ☺

# HAIL-CAESAR LibGeoDecomp port

- Define custom **Cell** class:
  - Contains all member data types for each grid cell
    - (e.g. double elevation, double water depth)
  - Must contain **update()** function
    - this is called by LibGeoDecomp during each time step
  - Need enum member type to distinguish between different cell types
    - (e.g. boundaries for application of boundary conditions)

- Define custom **Initializer** class:
  - Must extend a suitable LibGeoDecomp base Initializer class
  - Should define a **grid()** function, and use LibGeoDecomp's coordinate system syntax to initialise all grid cells (for serial execution) or only those cells in each rank's subgrid (for parallel execution)

# HAIL-CAESAR LibGeoDecomp port

- Declare an instance of a suitable LibGeoDecomp **Simulator** (serial, parallel, …) and pass it instances of your custom **Initializer** and a suitable **LoadBalancer**
- Commit LibGeoDecomp’s internal MPI Typemaps to MPI_COMM_WORLD by calling **initializeMaps()**
- Generate an MPI Typemap for your Cell class, and also commit *this* to MPI_COMM_WORLD
  - see next slide
- Add any **Writer**s to your **Simulator,** then let it run

# LibGeoDecomp and Typemap Generation

- If you want to run in parallel with MPI you *must* generate code and a header file describing an MPI Typemap for your custom Cell class
  - This must follow LibGeoDecomp's conventions
- Use doxygen and scripts supplied by LibGeoDecomp (in tools/typemapgenerator):
  - Make sure your Cell class declares Typemaps as a friend class
  - Run doxygen in your application dir to generate xml for your code
  - Run typemapgenerator.rb (Ruby) script that parses xml and writes typemaps.h & .cpp
  - Make sure these are compiled and included when you build your code

# Using LibGeoDecomp

- Heavily templated, multi-layered abstractions
  - Not easy to understand how everything fits together
- API documentation available as reference (but not a good starting point)
- Mini-application examples and unit tests help
  - These only cover a few usage scenarios / functionality aspects
  - Doing anything slightly different, needed to port existing applications, immediately requires understanding a lot of the underlying interlocking complexity

# Using LibGeoDecomp

- MPI Typemap generator
  - Not obvious from outset that needed!  (discoverability curve)
  - Encountered erronous typemap generation for enums (causing mini-application example code not to work) – found workaround
- Not straightforward to efficiently read in and initialise parallel simulation with real elevation data (DEM file)
  - Each rank could read same file, but for large numbers of ranks this will hit the filesystem hard serialising on single file, bottlenecking/throttling the application strongly – need workaround
  - Solution: read in file on rank 0 to initialise whole grid, write to file as MPI IO snapshot, read in MPI IO snapshot to initialise each subgrid in parallel

# Results & follow on work

- Scaling results on ARCHER for realistic DEMs of varying resolutions/sizes to follow in eCSE report

- Ported HAIL-CAESAR LibGeoDecomp code will be available on GitHub (http://dvalts.io/HAIL-CAESAR/ or linked to from there and from ARCHER website)

- Simple synthetic test cases not available in original HAIL-CAESAR provide valuable debugging tool and insight into LISFLOOD algorithm (extendable for erosion and additional processes)

# Follow on work

- Introducing parallel netCDF-based IO in LibGeoDecomp and thereby into HAIL-CAESAR
  - Paves the way for efficient ingest of self-describing high-resolution data, for initialisation and for ‘steering’ by feeding HPC simulation live flooding data to improve short-term acute forecasting
  - Equally enables efficient periodic output and storage of quantities (time series) of interest
  - Makes parallel netCDF IO functionality available to other application developers using LibGeoDecomp