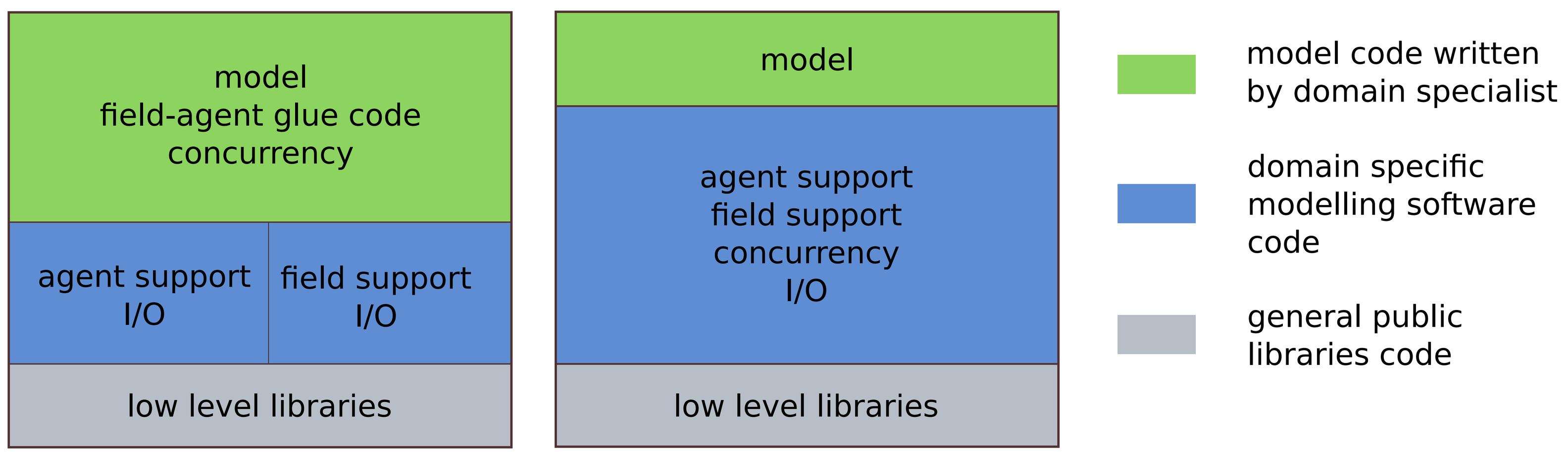
Improving the scalability of an environmental modelling framework to allow for large-scale high-resolution geosimulations

Universiteit Utrecht
Faculty of Geosciences

Oliver Schmitz, Kor de Jong and Derek Karssenberg

Introduction

The high computational requirements for stochastic spatio-temporal modelling, and an increasing demand to run models over large areas at high resolution, e.g. in global hydrological modelling or epidemiology, require an optimal use of available, heterogeneous computing resources. Domain-specific modelling software used by environmental scientists, however, often do not provide built-in capabilities to distribute model runs over the compute nodes of a supercomputer. We propose to enhance the PCRaster model building framework with built-in capabilities to run models on various hardware platforms, resulting in hardware scalable models that can be constructed by environmental modellers.



The PCRaster modelling framework

The PCRaster modelling framework
- Is targeted at the development of spatio-temporal models
- Fast model development and execution
- Scripting environments: PCRcalc and Python
- Rich set of model building blocks for manipulating raster maps
- Framework for stochastic spatio-temporal model building
- Framework for data assimilation
- Tool for visualisation of spatio-temporal stochastic data
- Runs on Linux, Microsoft Windows and Apple OS X
- Can be downloaded for free and is open source

Stochastic spatio-temporal modelling

Model

$$\mathbf{z}_t = f(\mathbf{z}_{t-1}, \mathbf{i}_t, \mathbf{p}) \quad \text{for all time steps t=1,...,T}$$

state variables　　inputs　　parameters
transition function

Solution scheme

for each n in Monte Carlo samples:
　for each t in time steps:

$$\mathbf{z}_t^{(n)} = f(\mathbf{z}_{t-1}^{(n)}, \mathbf{i}_t^{(n)}, \mathbf{p}^{(n)})$$

Building blocks

```
discharge = kinematic(flowDir,precipitation,..)
```

result map　　spatial function　　input maps

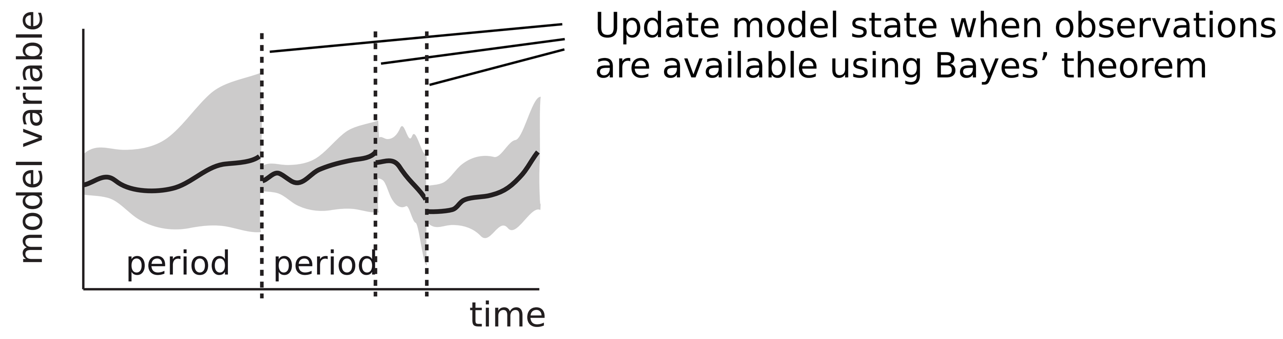
Solution framework (Python)

```
from pcraster import *
from pcraster.framework import *

class SnowModel(DynamicModel, MonteCarloModel):
  def __init__(self):
    ...
  def premcloop(self):
    dem = self.readmap("dem")
    self.ldd = lddcreate(dem, ...)
    ...
  def initial(self):
    self.snow = scalar(0)
    ...
  def dynamic(self):
    runoff = accuflux(self.ldd, rain)
    self.report(runoff, "q")
    ...
  def postmcloop(self):
    mcpercentiles("q",percentiles,..)
```

- premcloop: sets constant variables and parameters
- initial: is run at t = 0 for each Monte Carlo sample
- dynamic: is run for each Monte Carlo sample and for each time step
- postmcloop: is run at end calculating sampling statistics over Monte Carlo samples

Data assimilation



Solution scheme

for each period in periods:
　for each n in Monte Carlo Samples:
　　for each t in period:

$$\mathbf{z}_t^{(n)} = f(\mathbf{z}_{t-1}^{(n)}, \mathbf{i}_t^{(n)}, \mathbf{p}^{(n)})$$

　evaluate Bayes' theorem

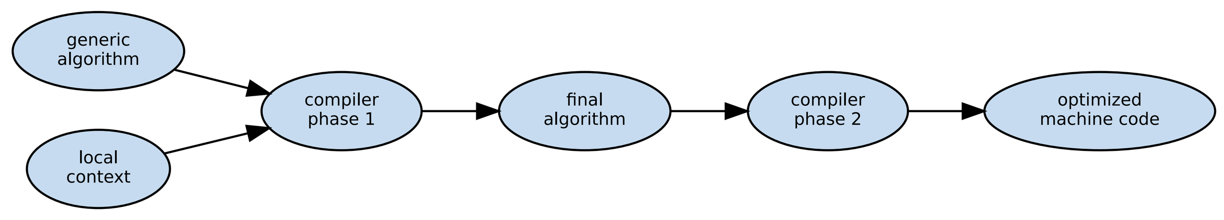
Solution framework (Python)

```
def suspend(self):
  self.report(self.snow, "s")
  ...
def updateWeight(self):
  sum = exp(maptotal(((obs - mod)**2)/
        (2.0 * (observedStd ** 2))))
  weight = exp(sum)
  return weight
  ...
def resume(self):
  self.read("s")
  ...
```

- suspend: store model state at end of period
- updateWeight: calculate weight of Monte Carlo sample required for solution of Bayes' equation and return to framework
- resume: read model state at start of next period

PCRaster on shared memory systems

A binding between PCRaster and Fern provides about 50 parallel local and focal operations. Fern is a is a highly generic C++ software library for raster processing that can be tailored to the configuration of a modelling framework.



Policy	Behaviour
execution	kind of parallelism
input no-data	how to handle no-data in input
output no-data	how to handle no-data in output
out-of-domain	how to test for out-of-domain values in input
out of range	how to handle out-of-range values in algorithm result

Configuration options per algorithm

```
template<
    template<typename> class OutOfDomainPolicy, // <--
    typename InputNoDataPolicy,
    typename OutputNoDataPolicy,
    typename ExecutionPolicy,
    typename Value,
    typename Result
>
void sqrt(
    InputNoDataPolicy const& input_no_data_policy,
    OutputNoDataPolicy& output_no_data_policy,
    ExecutionPolicy& execution_policy,
    Value const& value,
    Result& result);
```

Excerpt of C++ implementation of square root

```
from pcraster import *
values = readmap("raster.map")
result = sqrt(values)
```

Operation as used by modeller

PCRaster on distributed memory systems

Algorithms that operate on an irregular topology, such as material transport over a local drainage network, require a decomposition into fine grained sets of concurrent tasks for efficient execution. These tasks will be connected with other tasks from multiple algorithms into a task-graph, and an external HPX runtime library executes all tasks both on shared and distributed memory systems.

Download and further information

http://www.pcraster.eu

https://github.com/geoneric/fern

http://stellar.cct.lsu.edu/

D. Karssenberg, O. Schmitz, P. Salamon, K. de Jong, and M. F. P. Bierkens. A software framework for construction of process-based stochastic spatio-temporal models and data assimilation. Environmental Modelling & Software, 25 (4):489–502, 2010. doi: 10.1016/j.envsoft.2009.10.004

M. P. de Bakker, K. de Jong, O. Schmitz, and D. Karssenberg. Design and demonstration of a data model to integrate agent-based and field-based modelling. Environmental Modelling & Software, 89:172–189, 2017. doi: 10.1016/j.envsoft.2016.11.016