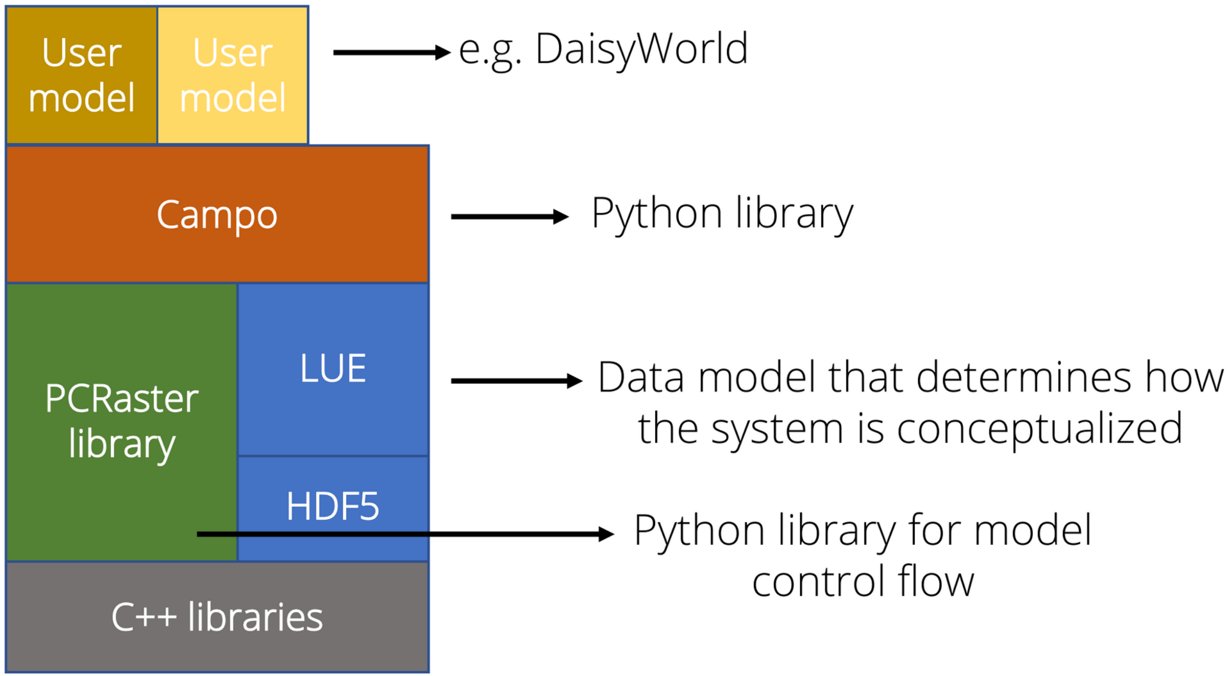
LUE tutorial

Oliver Schmitz & Judith A. Verstegen

Kor de Jong, Derek Karssenberg

LUE


User model
User model
e.g. DaisyWorld
Campo
Python library
PCRaster library
LUE
HDF5
Data model that determines how the system is conceptualized
Python library for model control flow
C++ libraries

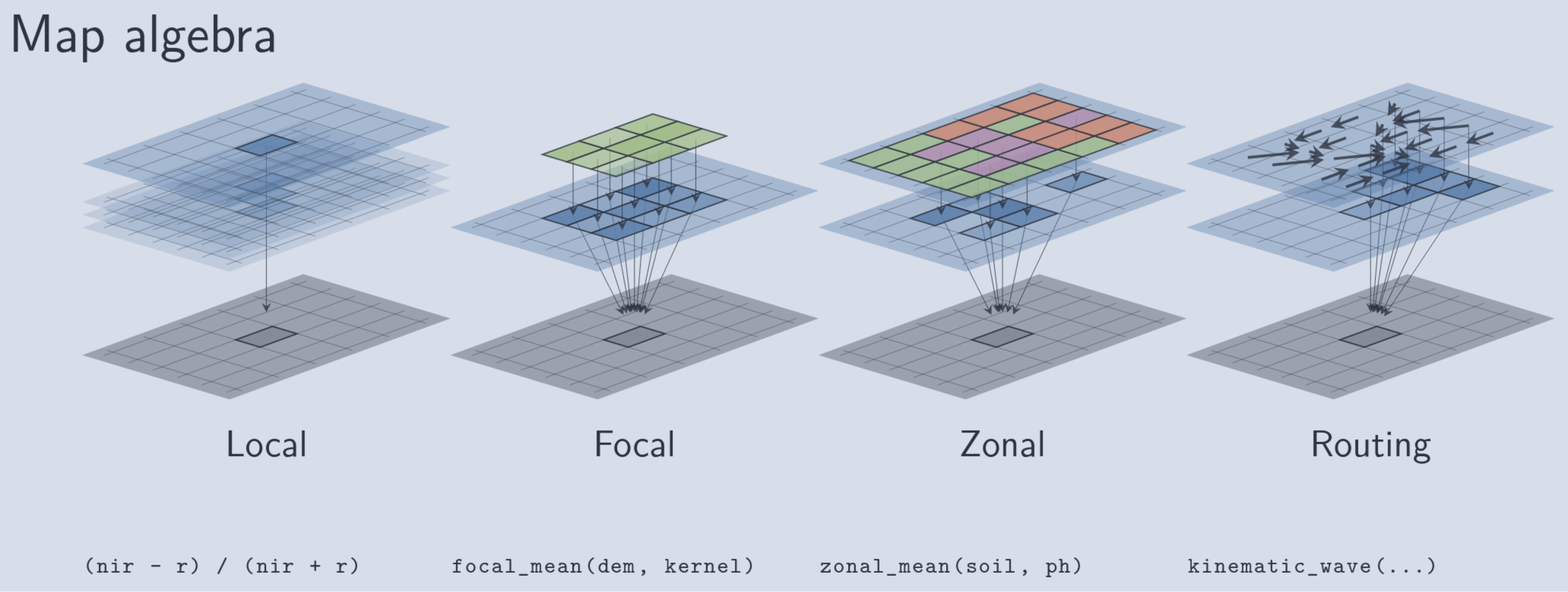
LUE

Modelling framework for simulating large geographical systems of agents and fields

- Modelling framework
  - Usable by software developers
  - Usable by model developers
- Large
  - No arbitrary limits
  - Model size → data set size & number of computations
  - Scalable over CPU cores and cluster nodes
- Systems of agents and fields
  - Integration of agent-based and field-based modelling
  - Single data model

LUE

- Field based modelling



LUE

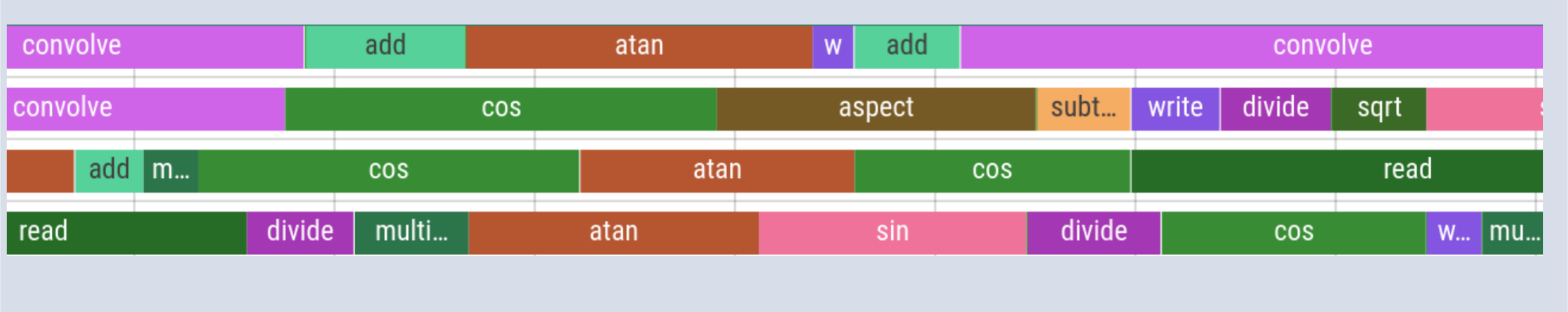
- Asynchronous many-tasks (AMT) approach:
  - Task: work to be executed on a CPU core
  - Sufficient amount of work
  - Independent order of execution

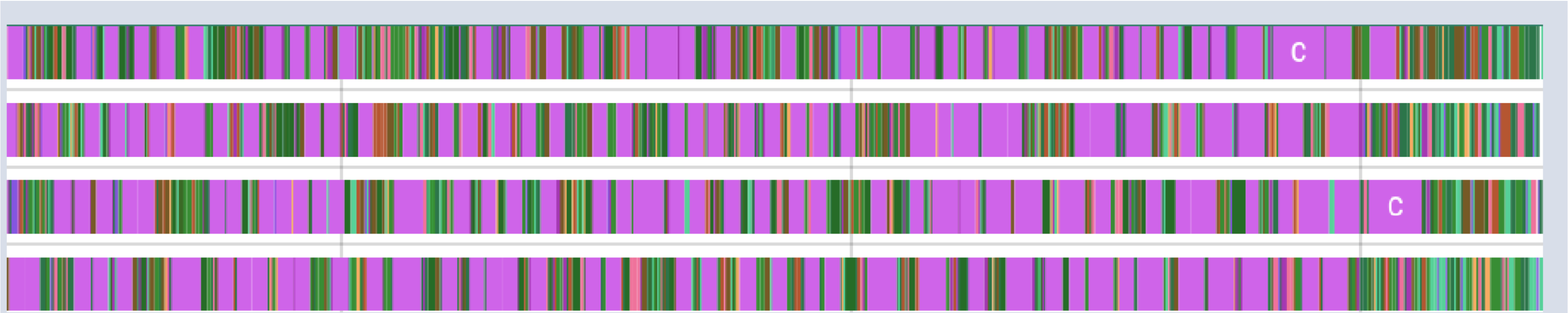
LUE

Possible order of execution:

LUE

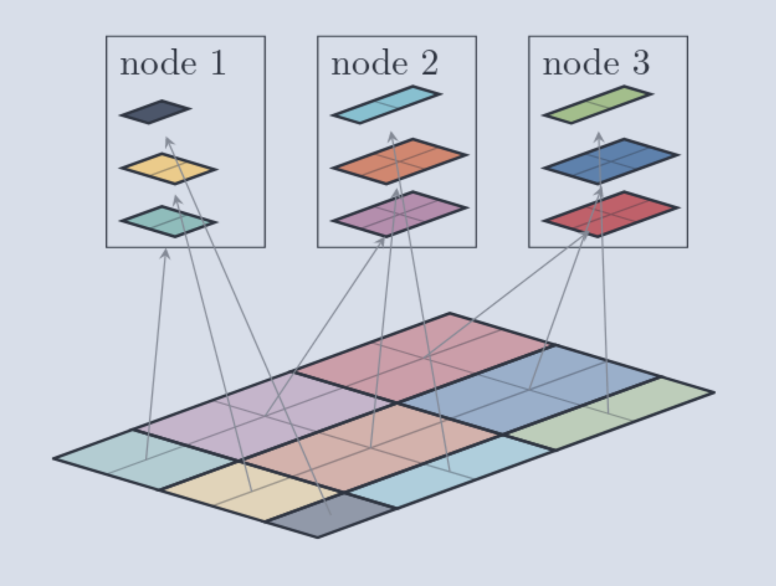
Possible execution order:





LUE

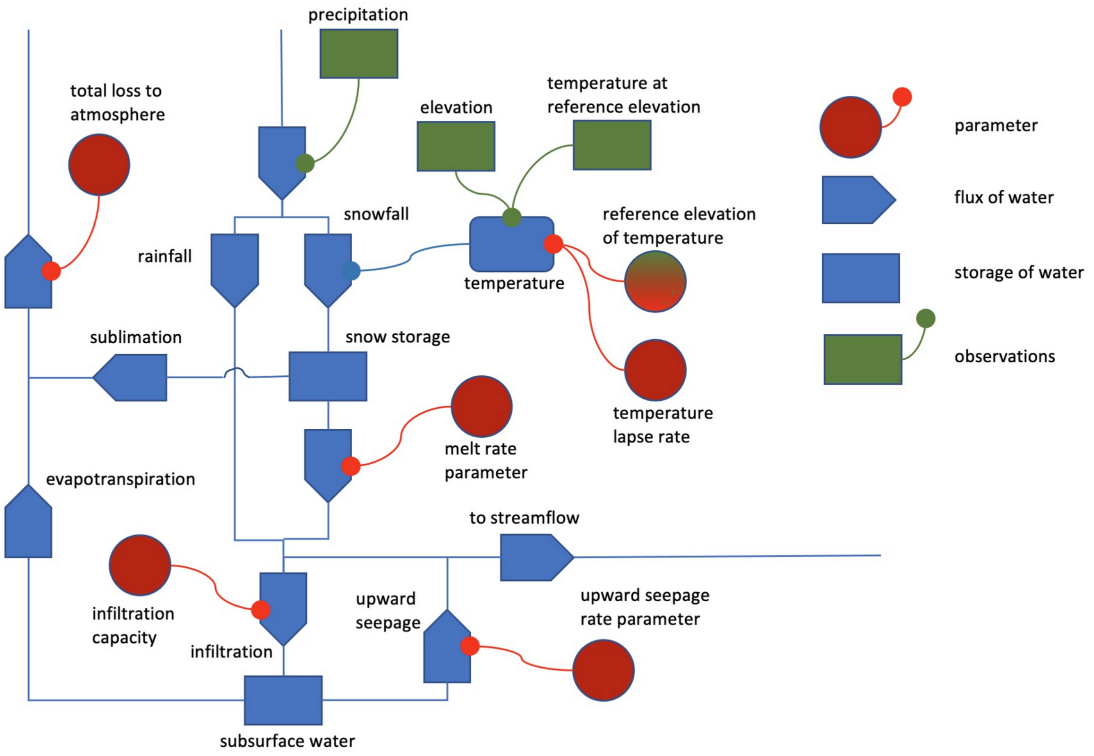
- Large computations:
  - Distribute work over CPU cores
  - Distribute work over cluster nodes



LUE

- Benefits:
  - Use a high-level API to define models (C++, Python)
  - Execute models faster
  - Execute larger models
  - No need to do the spatial domain partitioning yourself
  - No need to write parallel algorithms yourself
  - Open source, free to use, installable on all major platforms

A hydroLUEgical model



Air temperature:

$$t = t_{obs} - l(e - e_{obs})$$

Potential infiltration:

$$a = m(\max(t, 0)) + s$$

A hydroLUEgical model

```
def simulate(self, timestep):
    precipitation = self.precipitation[timestep - 1]
    observed_temperature = self.observed_temperature[timestep - 1]
    temperature = observed_temperature - self.temperature_correction

    freezing = temperature < 0.0
    snowfall = lfr.where(freezing, precipitation, 0.0)
    rainfall = lfr.where(~freezing, precipitation, 0.0)

    self.snow = self.snow + snowfall

    potential_melt = lfr.where(
        ~freezing, temperature * self.melt_rate_parameter, 0.0
    )
    actual_melt = minimum(self.snow, potential_melt)

    self.snow = self.snow - actual_melt

    # Sublimate first from atmospheric loss
    sublimation = minimum(self.snow, self.atmospheric_loss)
    self.snow = self.snow - sublimation

    # Potential evapotranspiration from subsurface water (m/day)
    potential_evapotranspiration = maximum(self.atmospheric_loss - sublimation, 0.0)

    # Actual evapotranspiration from subsurface water (m/day)
    evapotranspiration = minimum(
        self.subsurface_water, potential_evapotranspiration
    )
```

$$t = t_{obs} - l(e - e_{obs})$$

A hydroLUEgical model

```python
def simulate(self, timestep):
    precipitation = self.precipitation[timestep - 1]
    observed_temperature = self.observed_temperature[timestep - 1]
    temperature = observed_temperature - self.temperature_correction

    freezing = temperature < 0.0
    snowfall = lfr.where(freezing, precipitation, 0.0)
    rainfall = lfr.where(~freezing, precipitation, 0.0)

    self.snow = self.snow + snowfall

    potential_melt = lfr.where(
        ~freezing, temperature * self.melt_rate_parameter, 0.0
    )
    actual_melt = minimum(self.snow, potential_melt)

    self.snow = self.snow - actual_melt

    # Sublimate first from atmospheric loss
    sublimation = minimum(self.snow, self.atmospheric_loss)
    self.snow = self.snow - sublimation

    # Potential evapotranspiration from subsurface water (m/day)
    potential_evapotranspiration = maximum(self.atmospheric_loss - sublimation, 0.0)

    # Actual evapotranspiration from subsurface water (m/day)
    evapotranspiration = minimum(
        self.subsurface_water, potential_evapotranspiration
    )
```

Same script for

- Notebook
- Workstation
- HPC

A hydroLUEgical model

Demo

LUE

- Current emphasis is on field-based operations
- Will replace PCRaster in Campo
- Research on agent-based operations