

**A high performance
computing
environment for
cardiac simulations.**

Ross McFarlane
post-Postgraduate Workshop 2009

Overview.

QUI

Distributed Memory Parallelisation

Optimisation

QUI.

Modular software package, written in C

Provides a selection of models of kinetics

Produces a range of output data

on-screen

snapshot images

data files

Reaction-diffusion equations.

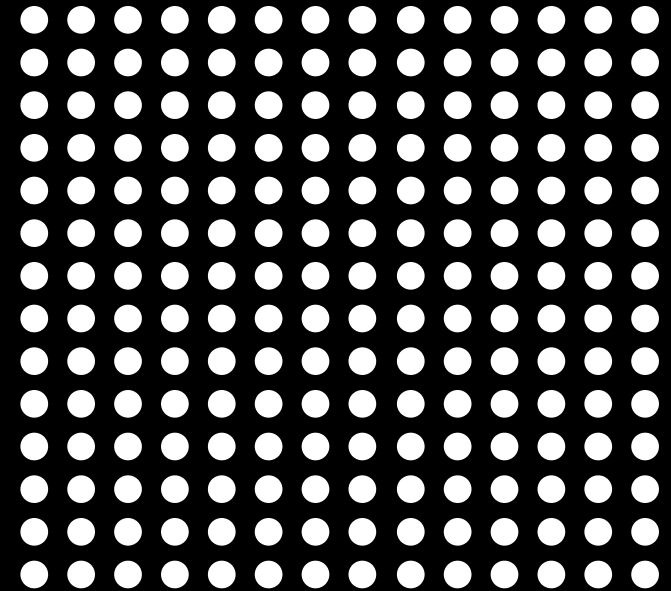
$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{f}(\mathbf{u}) + \mathbf{D} \Delta \mathbf{u}$$

u.

u, f $\in \mathbb{R}^l$, $l \geq 2$

u(x, y, z, t)

Simulation medium.

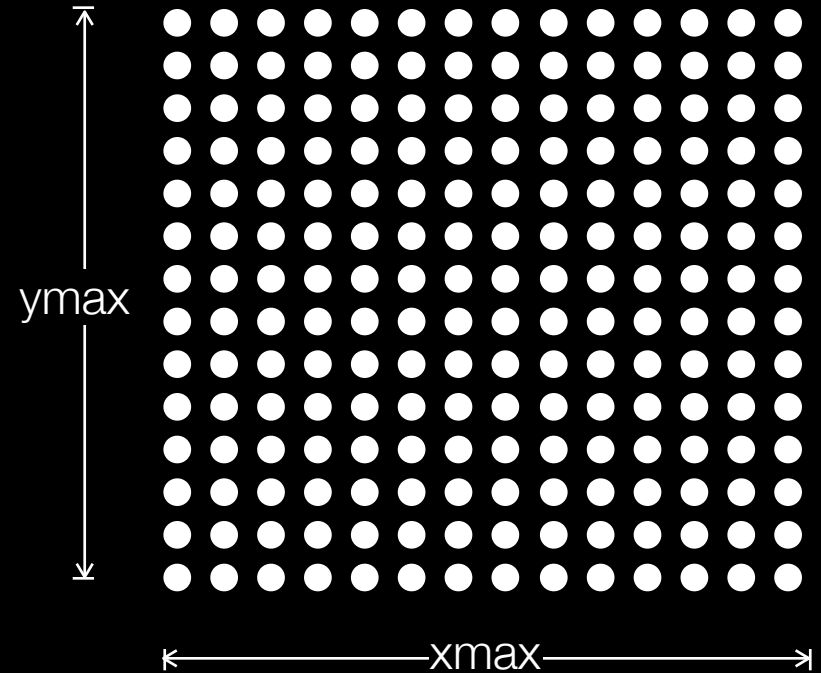


Simulation medium.

4-dimensional:

3-dimensional mesh

v_{\max} *dynamic variables* at
each point

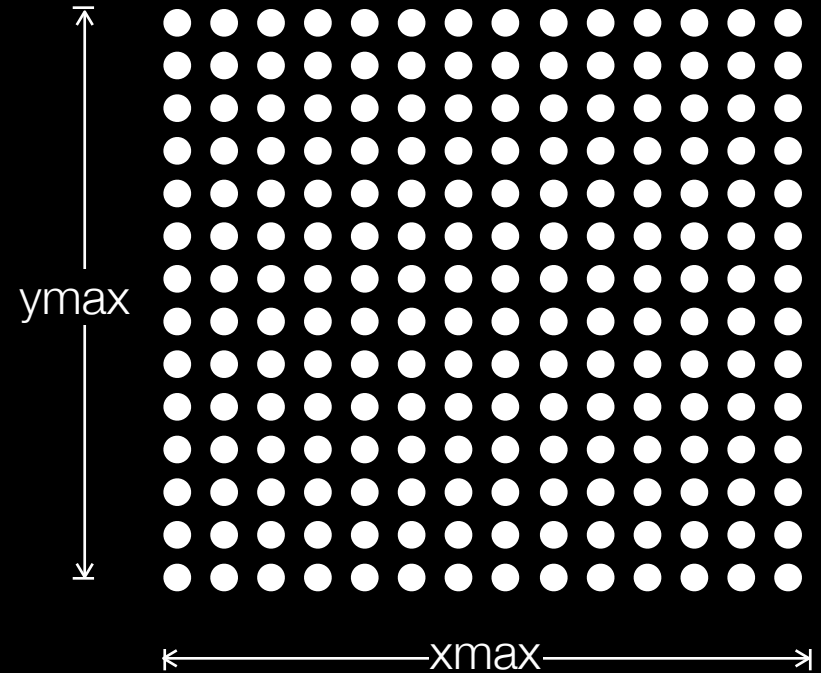


Simulation medium.

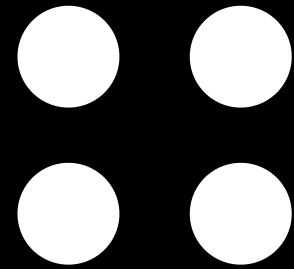
4-dimensional:

3-dimensional mesh

v_{\max} *dynamic variables* at
each point

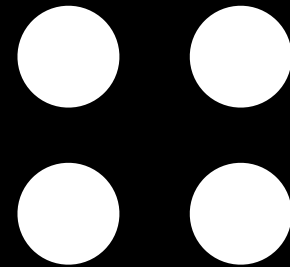


New.



New.

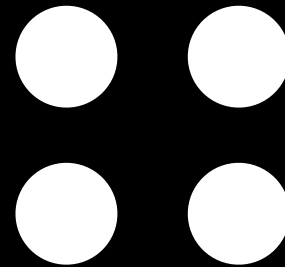
**One-dimensional
array**



New.

**One-dimensional
array**

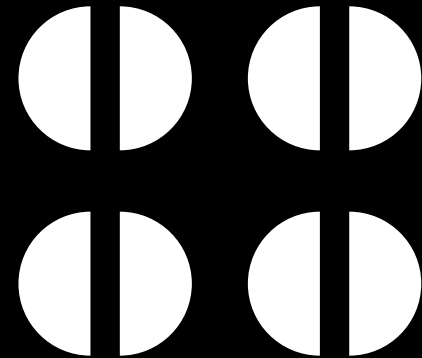
**ind() function
maps (x,y,z,v)
coordinates to
indices of New**



New.

**One-dimensional
array**

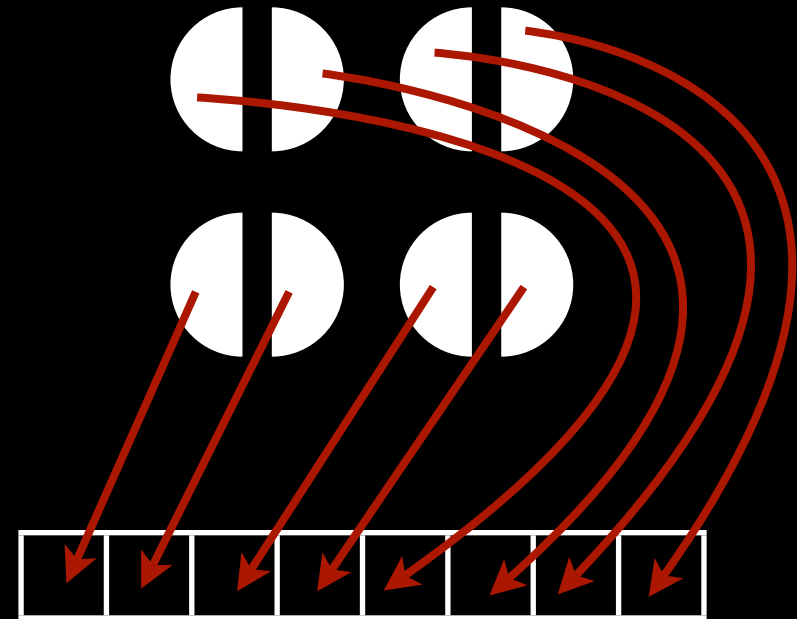
**ind() function
maps (x,y,z,v)
coordinates to
indices of New**



New.

**One-dimensional
array**

**ind() function
maps (x,y,z,v)
coordinates to
indices of New**

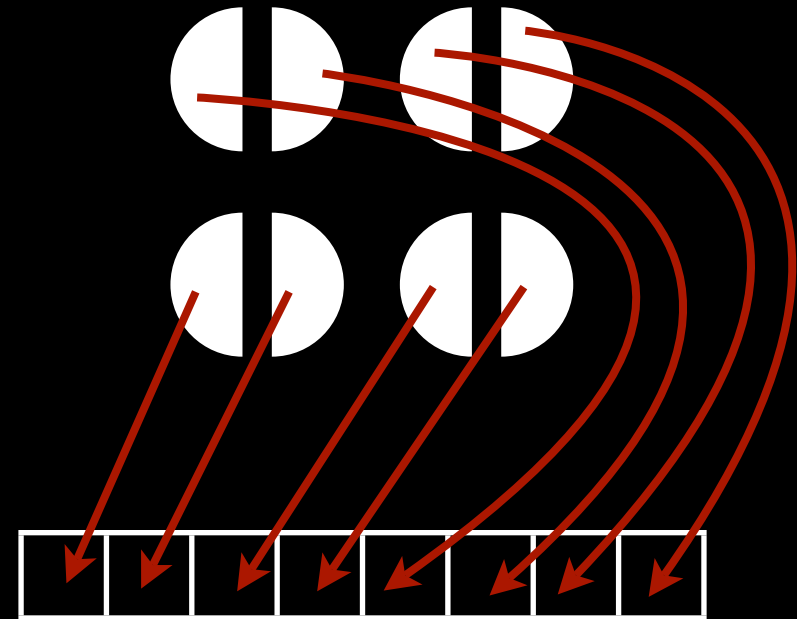


New.

**One-dimensional
array**

**ind() function
maps (x,y,z,v)
coordinates to
indices of New**

1 timestep



QUI devices.



QUI devices.

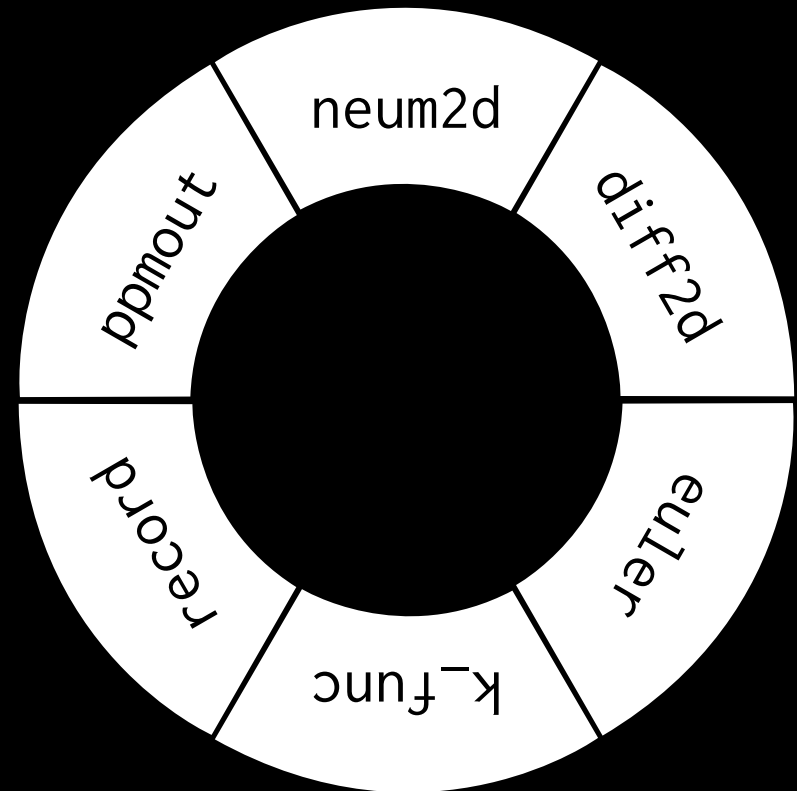
A device is a unit of functionality



QUI devices.

A device is a unit of functionality

Devices take turns to operate, in the *ring of devices*

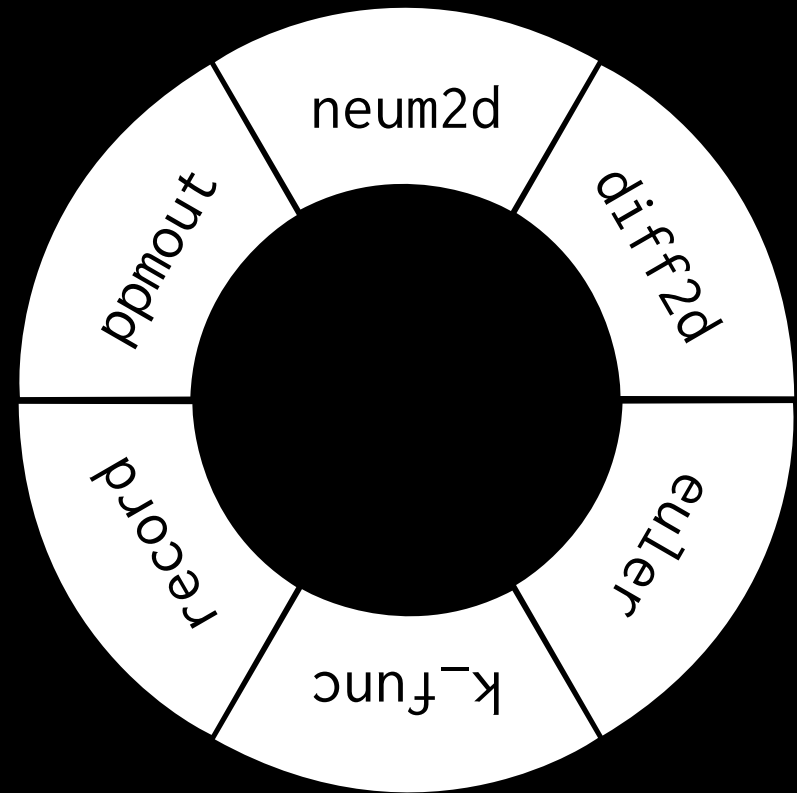


QUI devices.

A device is a unit of functionality

Devices take turns to operate, in the *ring of devices*

Each rotation is one timestep



QUI workflow.

```
state xmax=5 ymax=5 zmax=5 vmax=2;

<std.qui>
<fhn.qui>

def real begin;
def real end;

def int printInterval 50;
def real print;

neum2d ;
diff2d v0=0 v1=[IV];
euler v1=2 ode=fhncub par={IV=@2} ;

k_func pgm={
  print = eq(mod(t,printInterval),0);
  end = gt(t,10);
};

record when= print file=mySimulation.rec;
ppmout ;
stop when=end;
end;
```

Script

QUI workflow.

```
state xmax=5 ymax=5 zmax=5 vmax=2;
<std.qui>
<fhn.qui>

def real begin;
def real end;

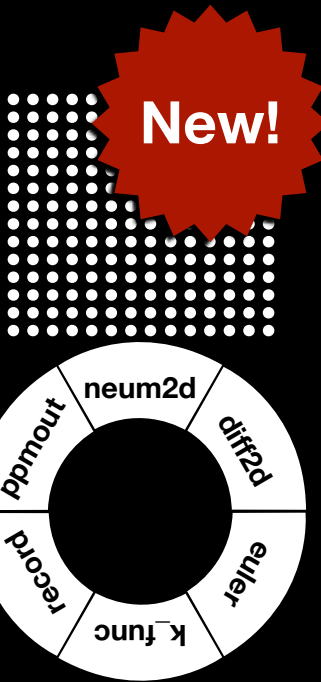
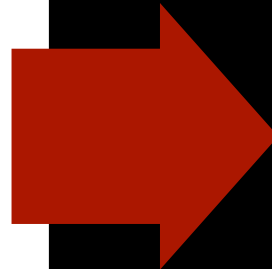
def int printInterval 50;
def real print;

neum2d ;
diff2d v0=0 v1=[IV];
euler v1=2 ode=fhncub par={IV=@2} ;

k_func pgm={
  print = eq(mod(t,printInterval),0);
  end = gt(t,10);
};

record when= print file=mySimulation.rec;
ppmout ;
stop when=end;
end;
```

Script



Create

QUI workflow.

```
state xmax=5 ymax=5 zmax=5 vmax=2;
<std.qui>
<fhn.qui>

def real begin;
def real end;

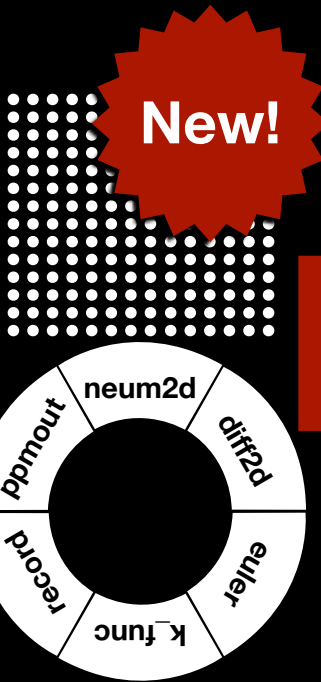
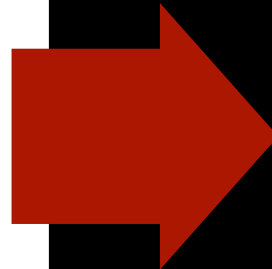
def int printInterval 50;
def real print;

neum2d ;
diff2d v0=0 v1=[IV];
euler v1=2 ode=fhncub par={IV=@2} ;

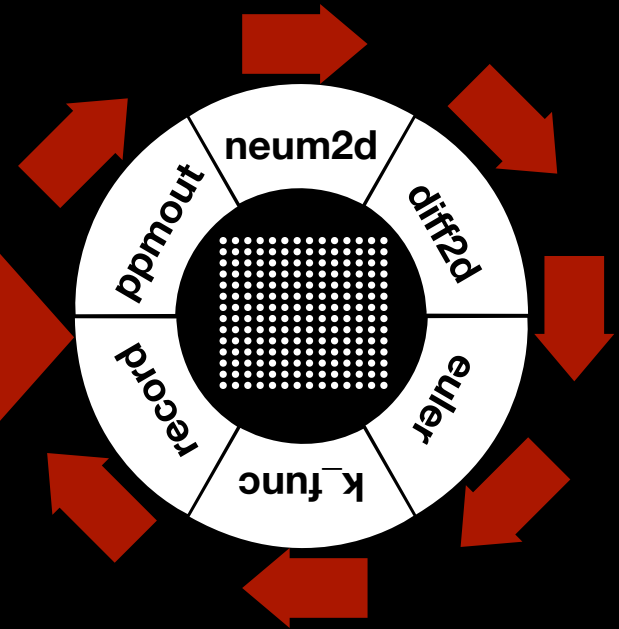
k_func pgm={
  print = eq(mod(t,printInterval),0);
  end = gt(t,10);
};

record when= print file=mySimulation.rec;
ppmout ;
stop when=end;
end;
```

Script



Create



Run

Core computation.

The euler Device

RHS Modules

Diffusion Devices

Boundary Conditions Devices



The eu1er device.



The euler device.

Computes the future state of the medium using
the *Euler Forward Method*:

$$\mathbf{u}^{t+1}(x, y, z) = \mathbf{u}^t + \Delta t \partial_t [\mathbf{f}(\mathbf{u}^t) + \mathbf{D}\Delta\mathbf{u}^t]$$



The euler device.

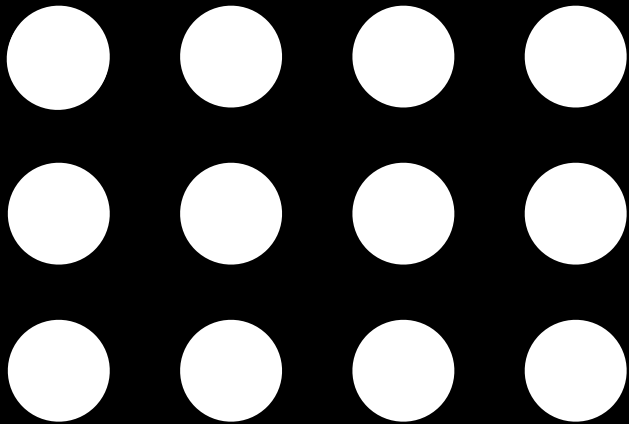
Computes the future state of the medium using the *Euler Forward Method*:

$$\mathbf{u}^{t+1}(x, y, z) = \mathbf{u}^t + \Delta t \partial_t [\mathbf{f}(\mathbf{u}^t) + \mathbf{D} \Delta \mathbf{u}^t]$$

Delegates computation of local kinetics ($\mathbf{f}(\mathbf{u}^t)$) to a Right Hand Side (RHS) module



The eu1er device.



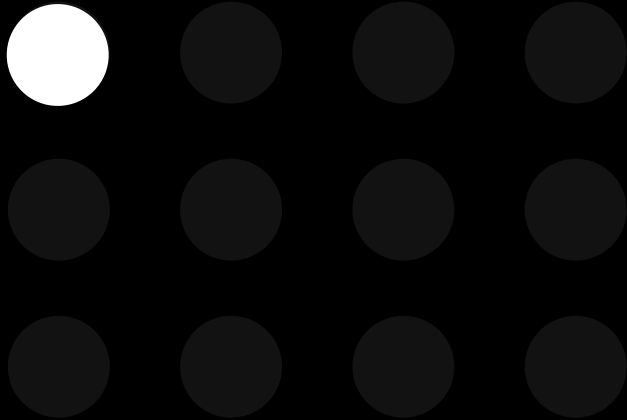
eu1er

RHS Module



The euler device.

$$\mathbf{u}^t(x, y, z)$$

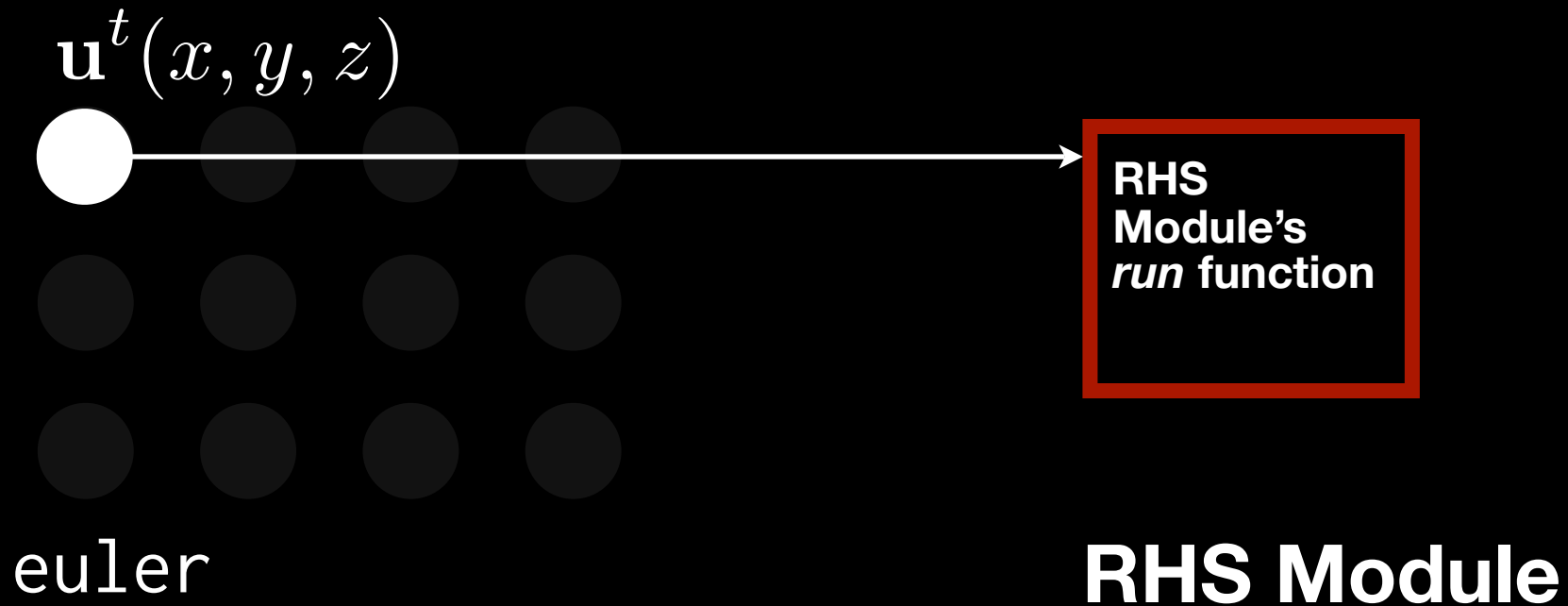


euler

RHS Module



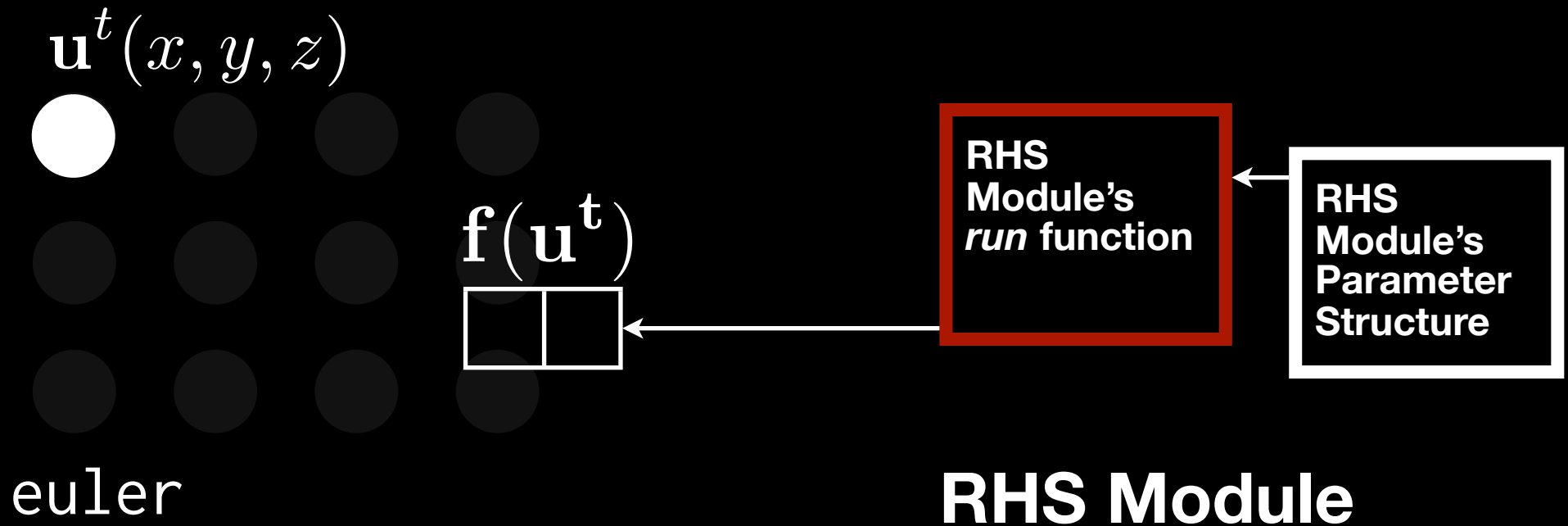
The euler device.



The euler device.

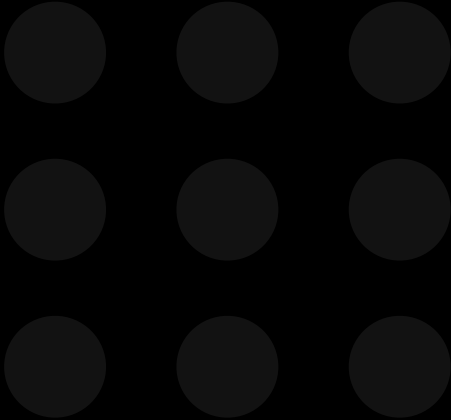


The euler device.

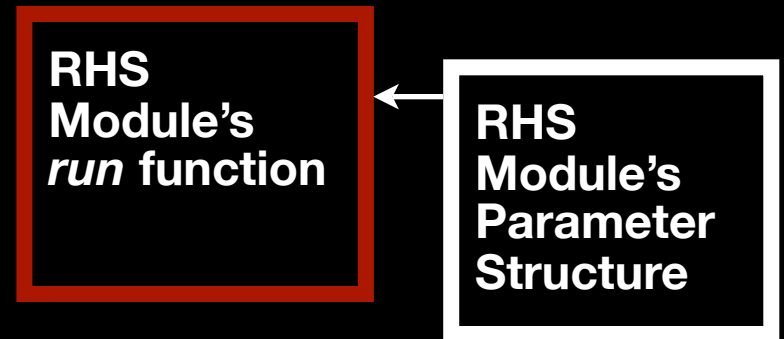


The euler device.

$u^t(x, y, z)$



euler



RHS Module



Diffusion.

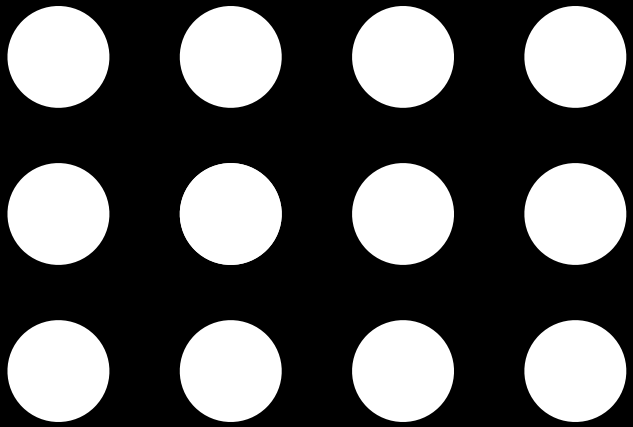
$$\mathbf{D}\Delta\mathbf{u}:$$

\mathbf{D} conductivity tensor

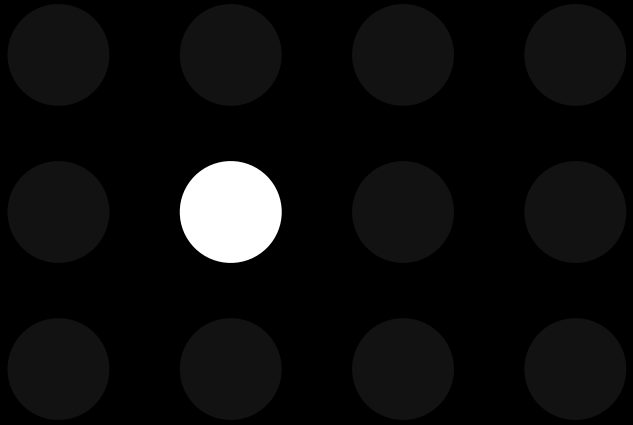
$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$



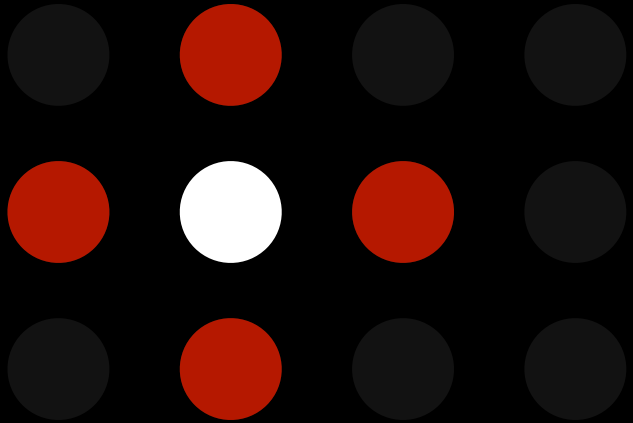
Diffusion.



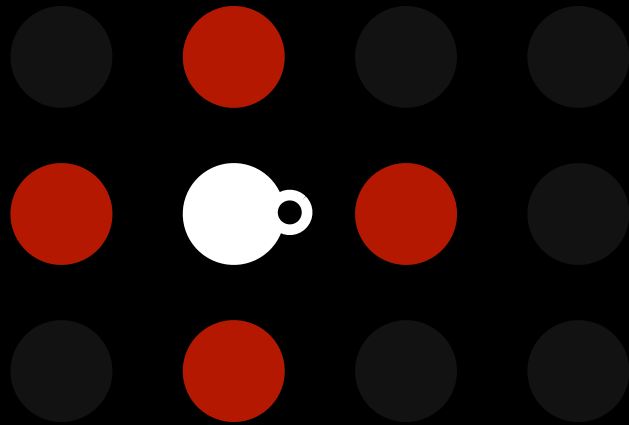
Diffusion.



Diffusion.



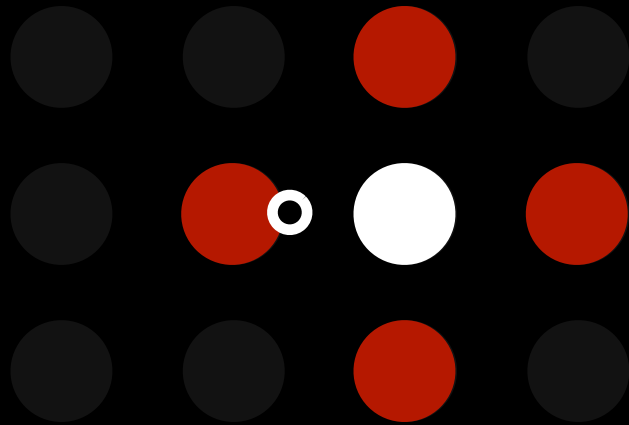
Diffusion.



The Laplacian value
is stored in an
additional dynamic
variable



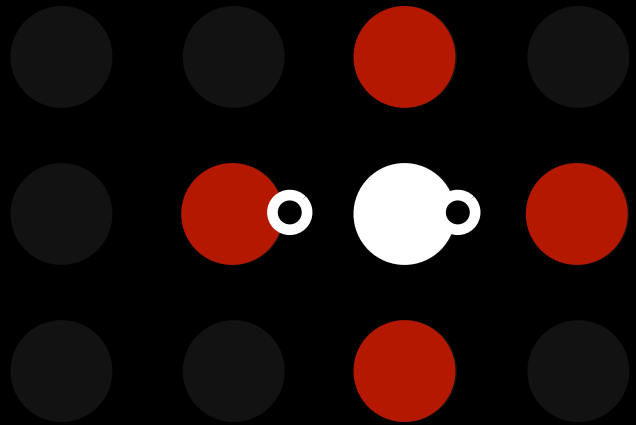
Diffusion.



The Laplacian value
is stored in an
additional dynamic
variable

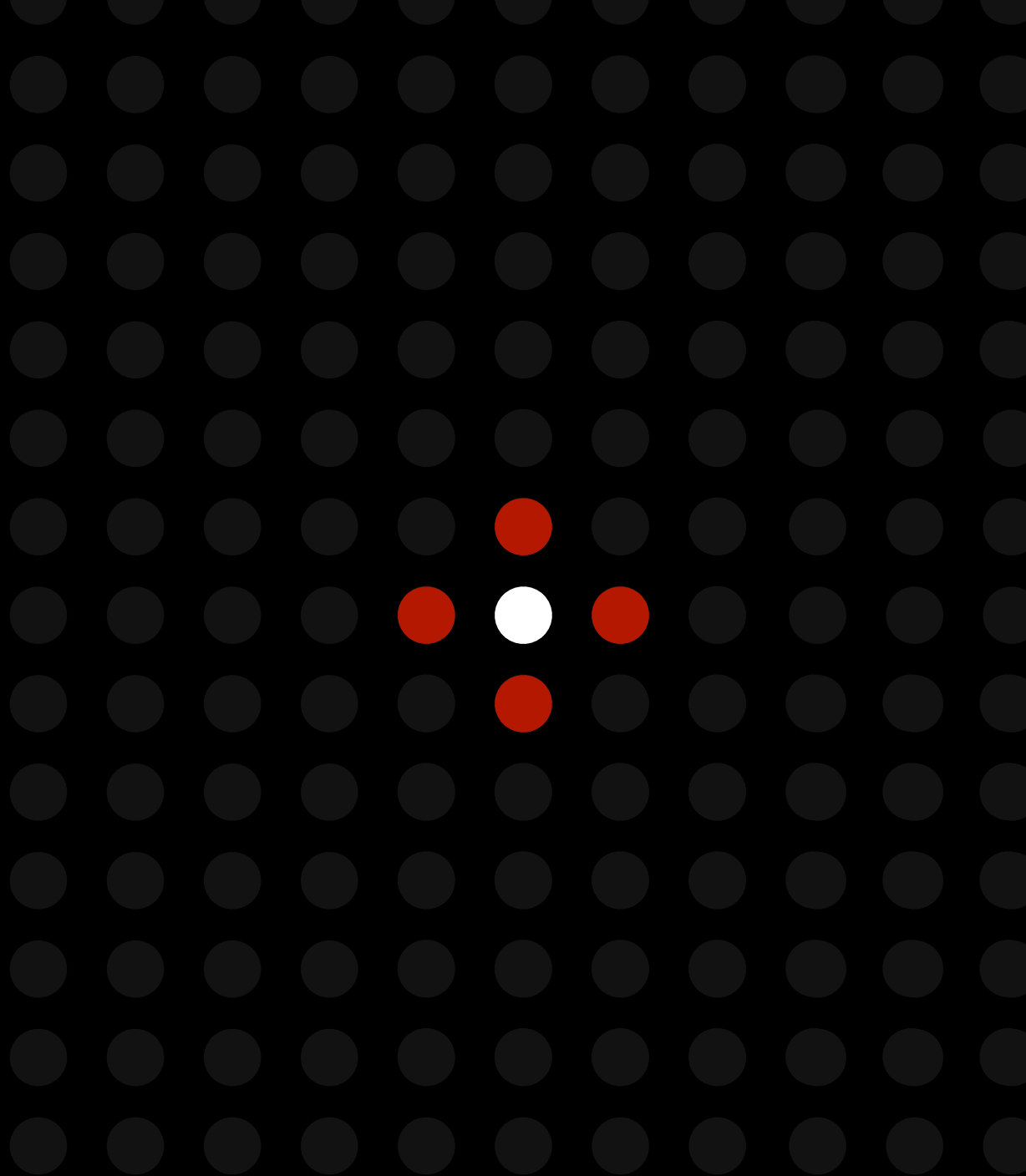


Diffusion.



The Laplacian value
is stored in an
additional dynamic
variable







?

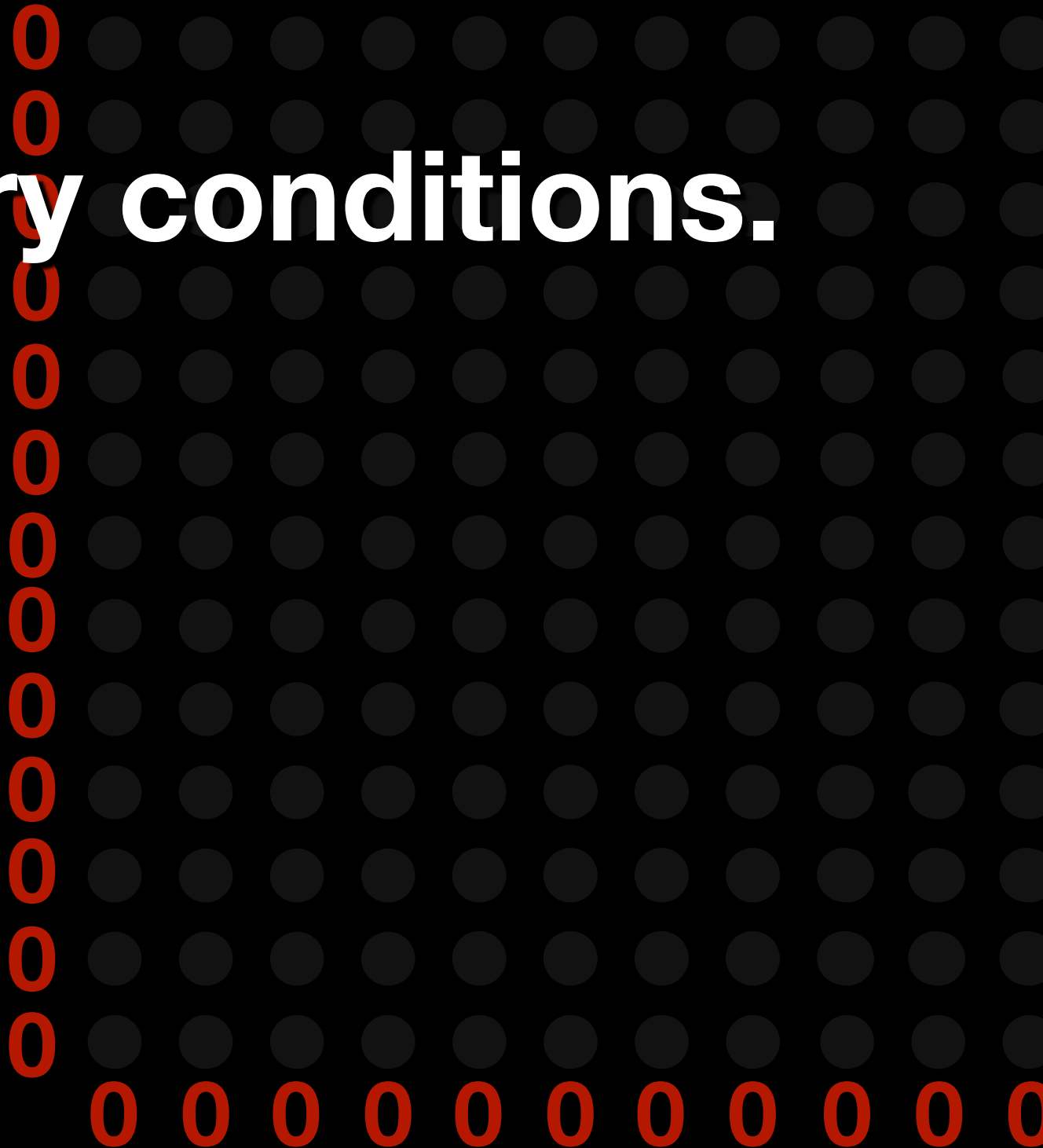


Boundary conditions.



Boundary conditions.

Fixed Value
Dirichlet



Boundary conditions.

Fixed Value

Dirichlet

Zero Flux

Neumann



Boundary conditions.

Fixed Value

Dirichlet

Zero Flux

Neumann



Boundary conditions.

Fixed Value

Dirichlet

Zero Flux

Neumann

No Boundary

Toroid



Boundary conditions.

Fixed Value

Dirichlet

Zero Flux

Neumann

No Boundary

Toroid



Distributed memory parallelisation.



Distributed memory parallelisation.

Multiple machines, each with local memory

Machines are connected by a network



Distributed memory parallelisation.

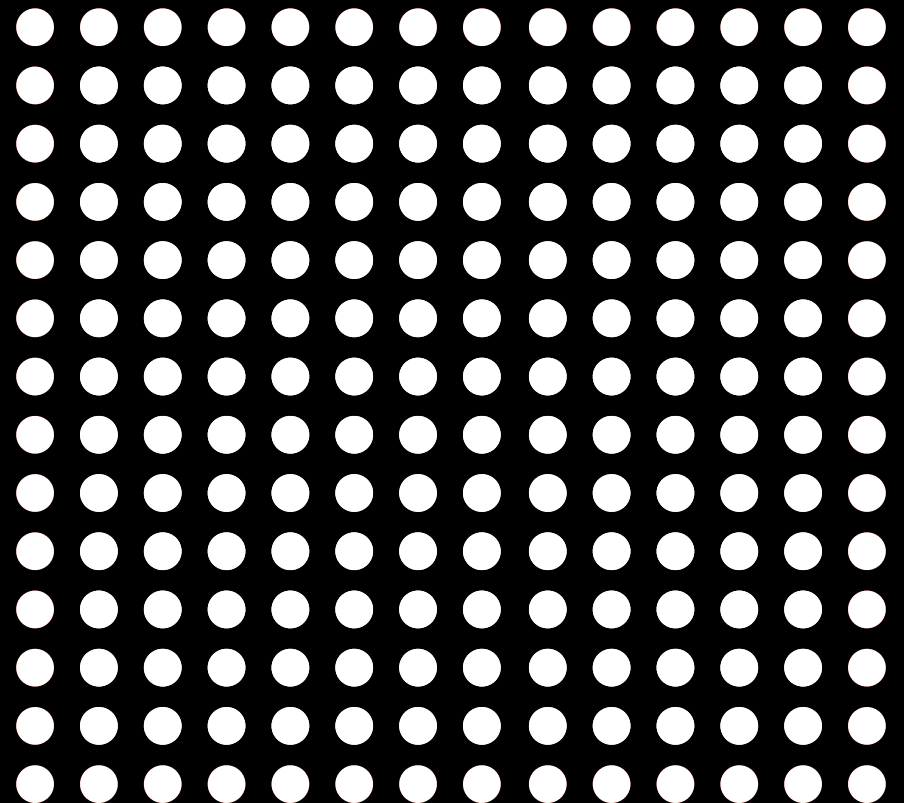
Multiple machines, each with local memory

Machines are connected by a network

The *full* program runs in every process

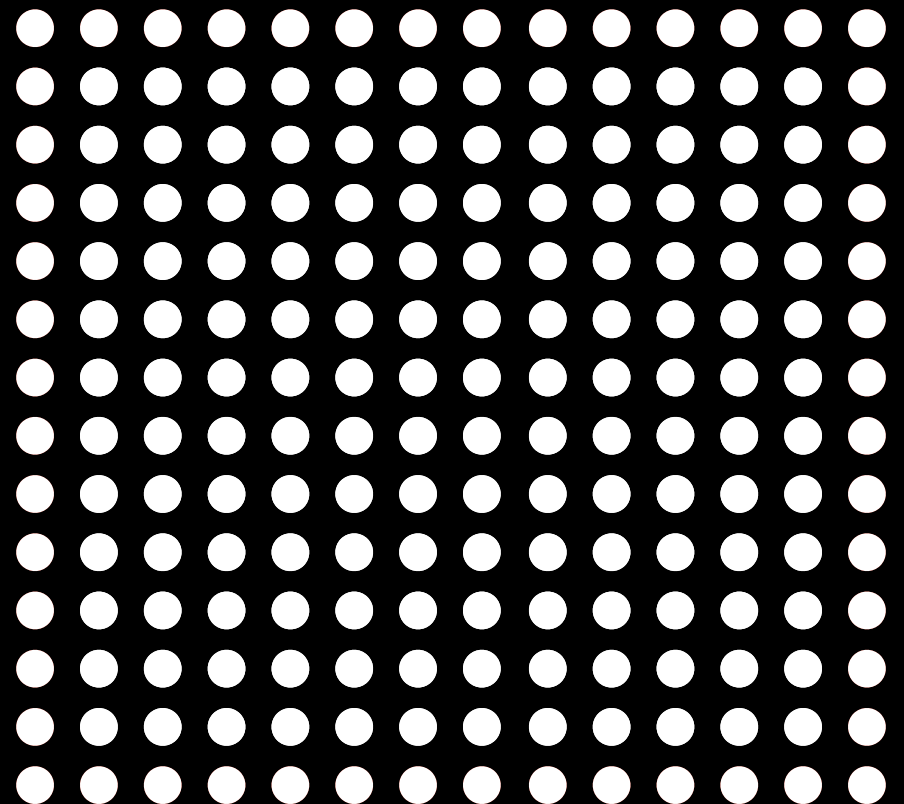


Domain decomposition.



Domain decomposition.

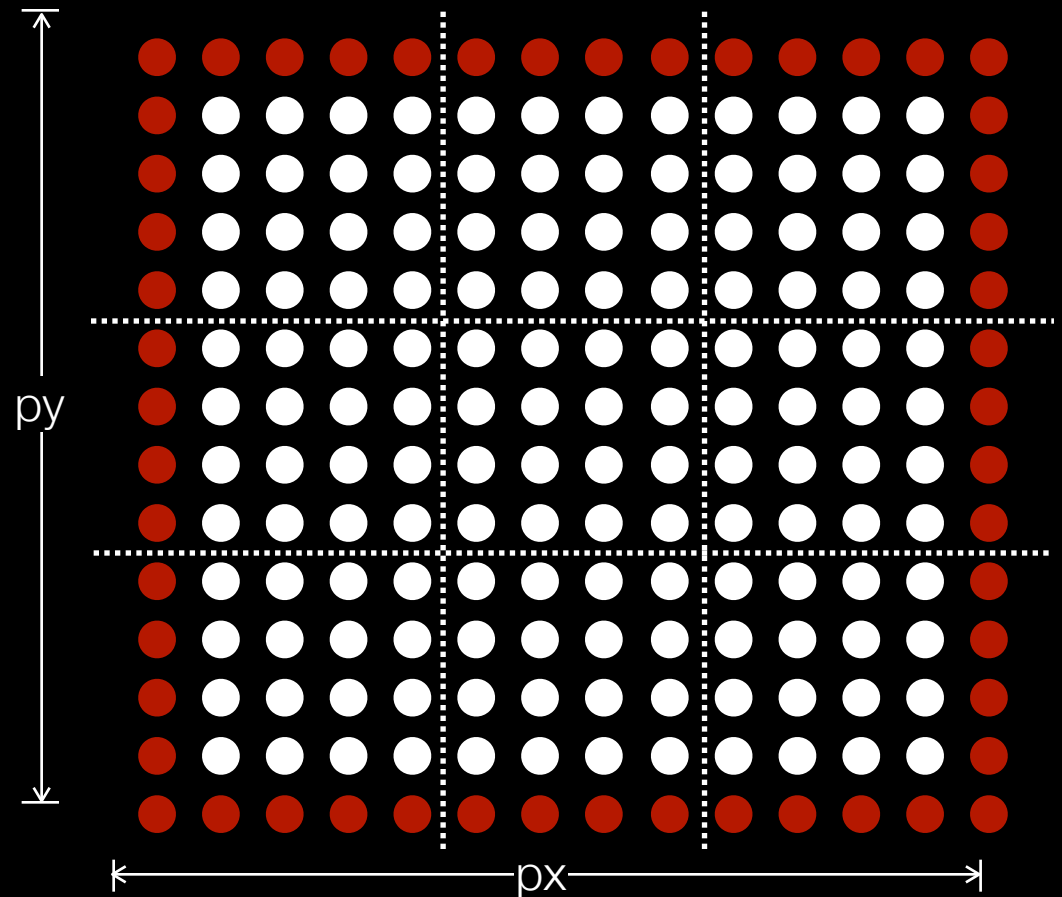
Supergrid



Domain decomposition.

Supergrid

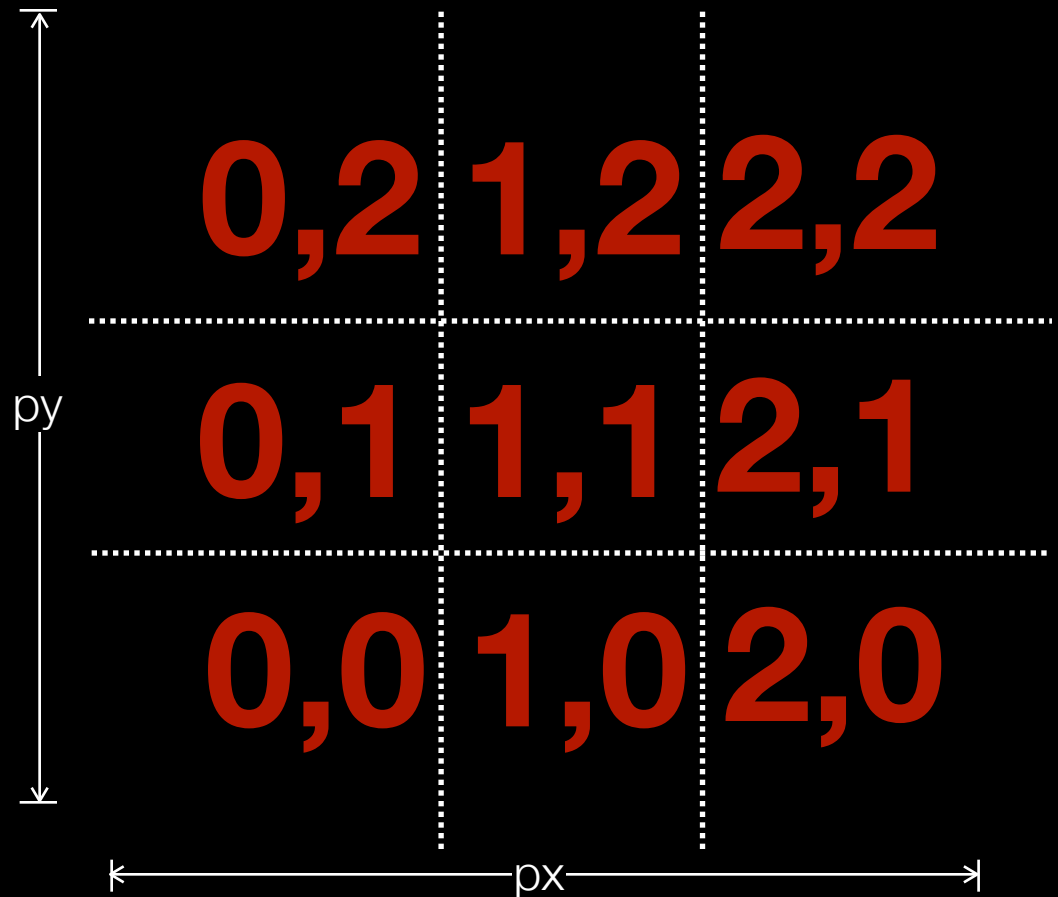
$p_x \times p_y \times p_z$



Domain decomposition.

Supergrid

$p_x \times p_y \times p_z$
superindices

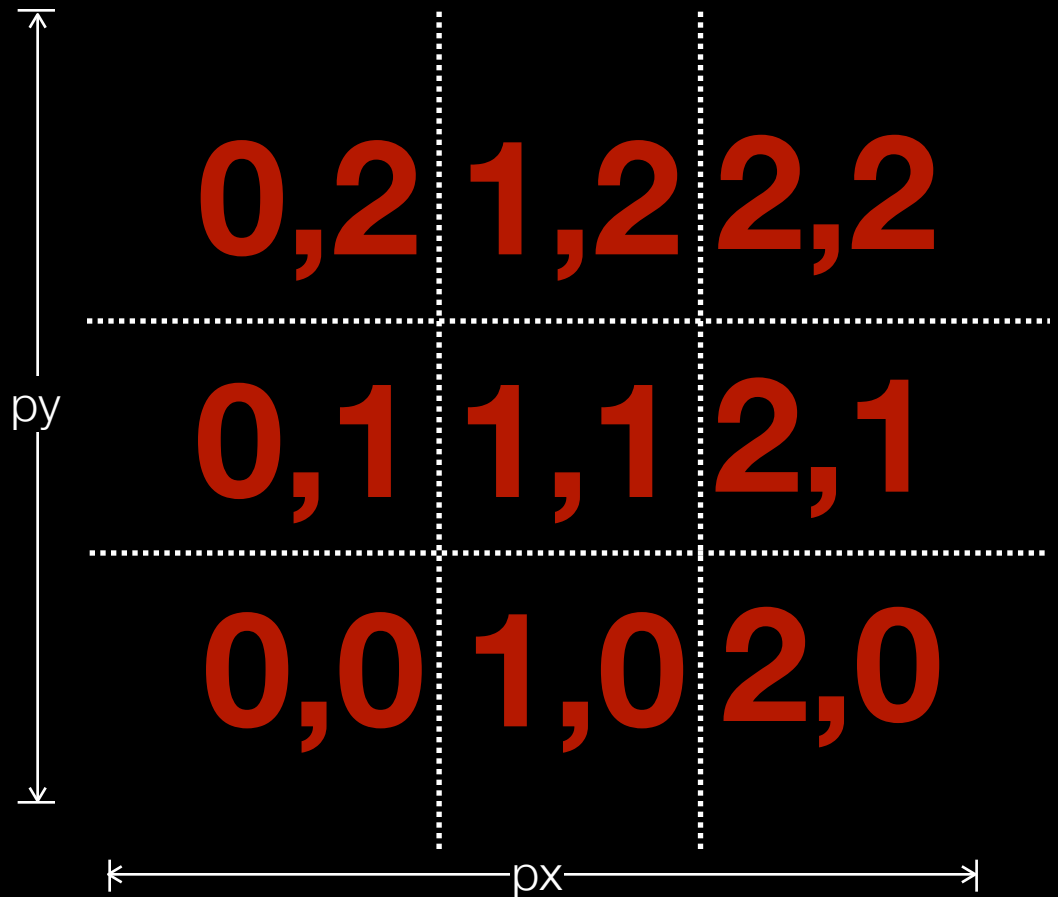


Domain decomposition.

Supergrid

$p_x \times p_y \times p_z$
superindices

Process Rank



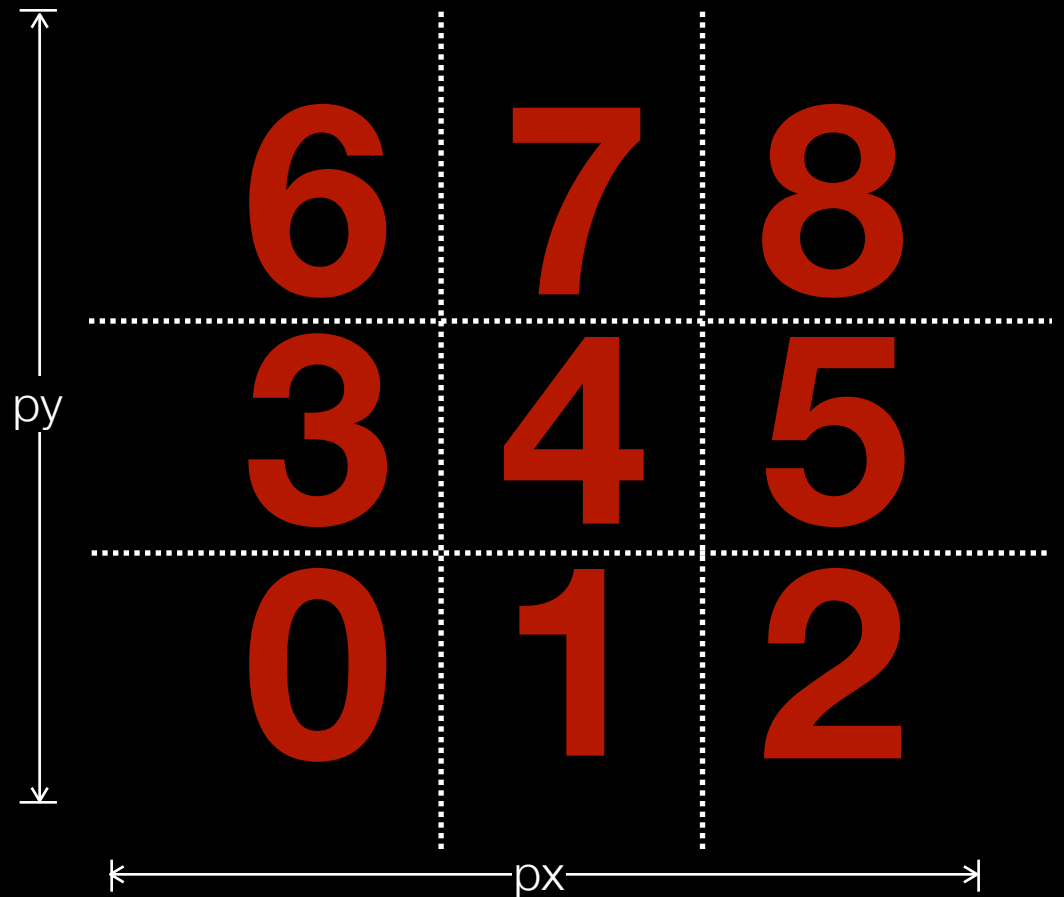
Domain decomposition.

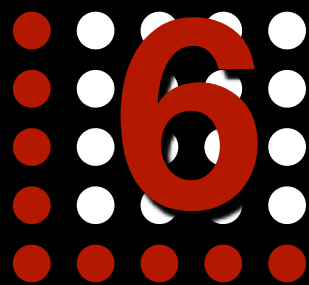
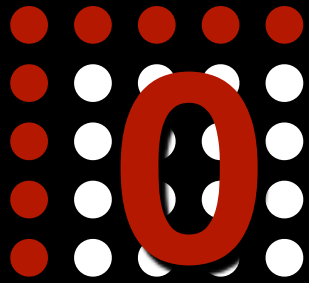
Supergrid

$p_x \times p_y \times p_z$
superindices

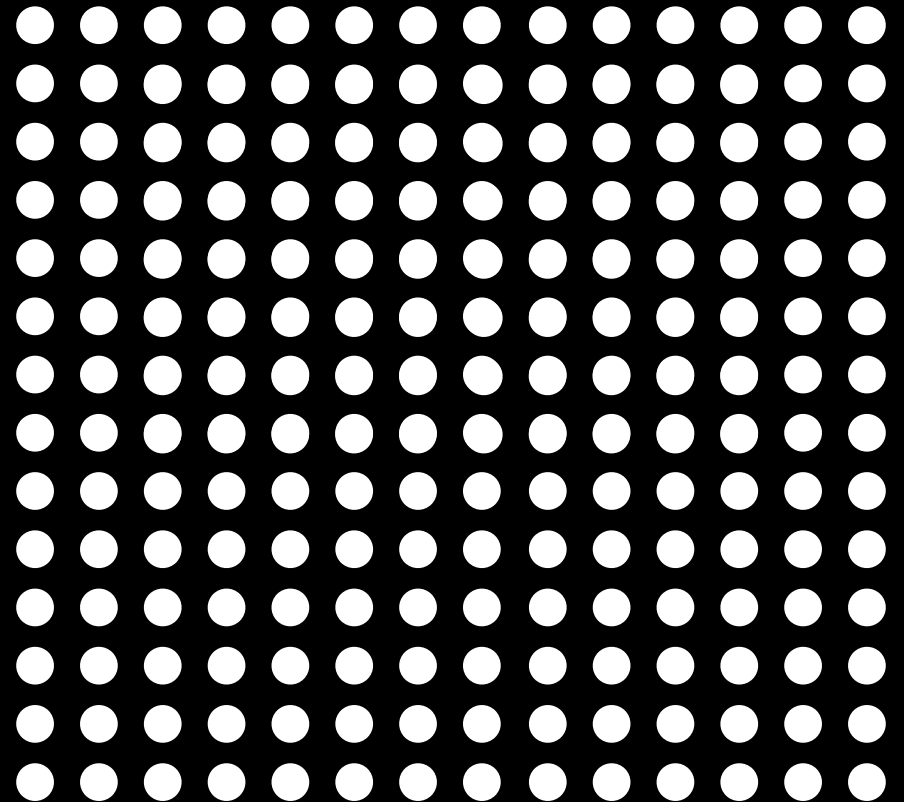
Process Rank

Map to superindices





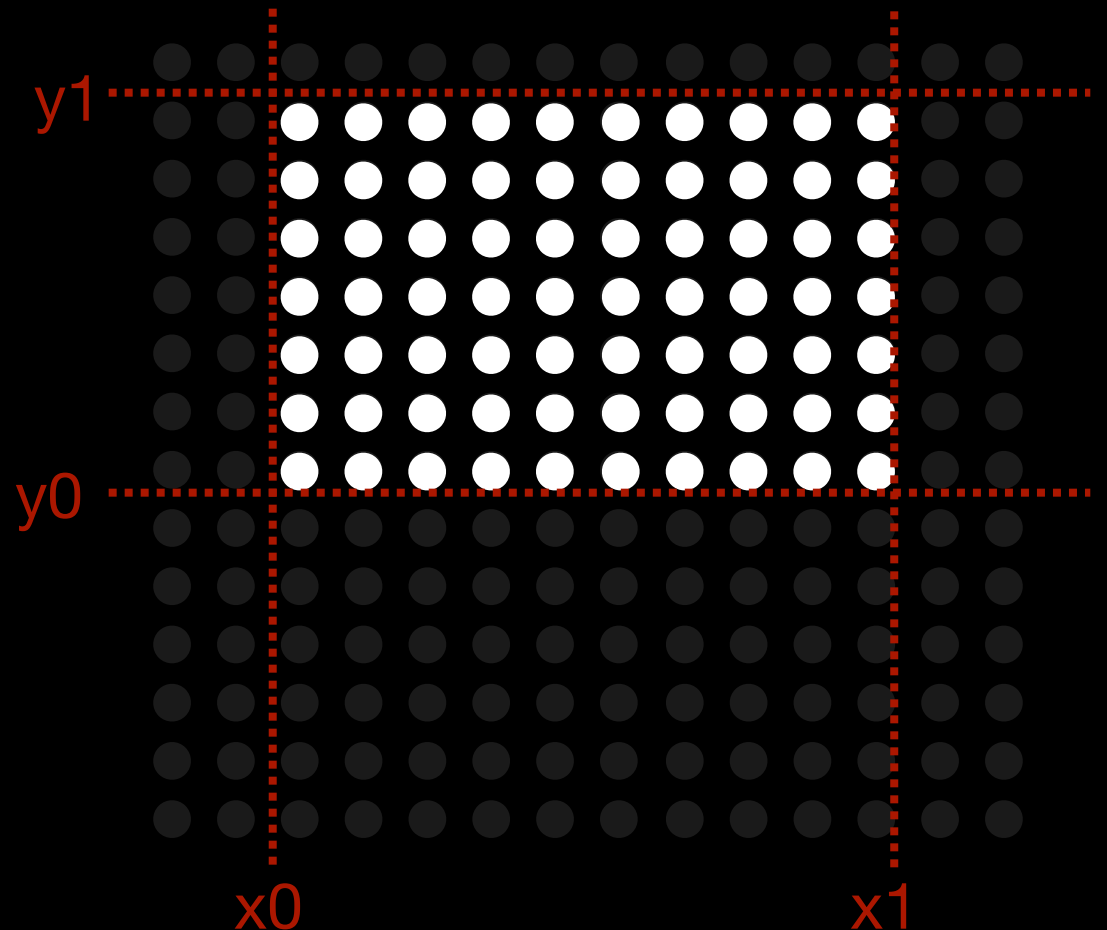
Device spaces.



Device spaces.

Each device operates within its *space*

Limits are specified by the user

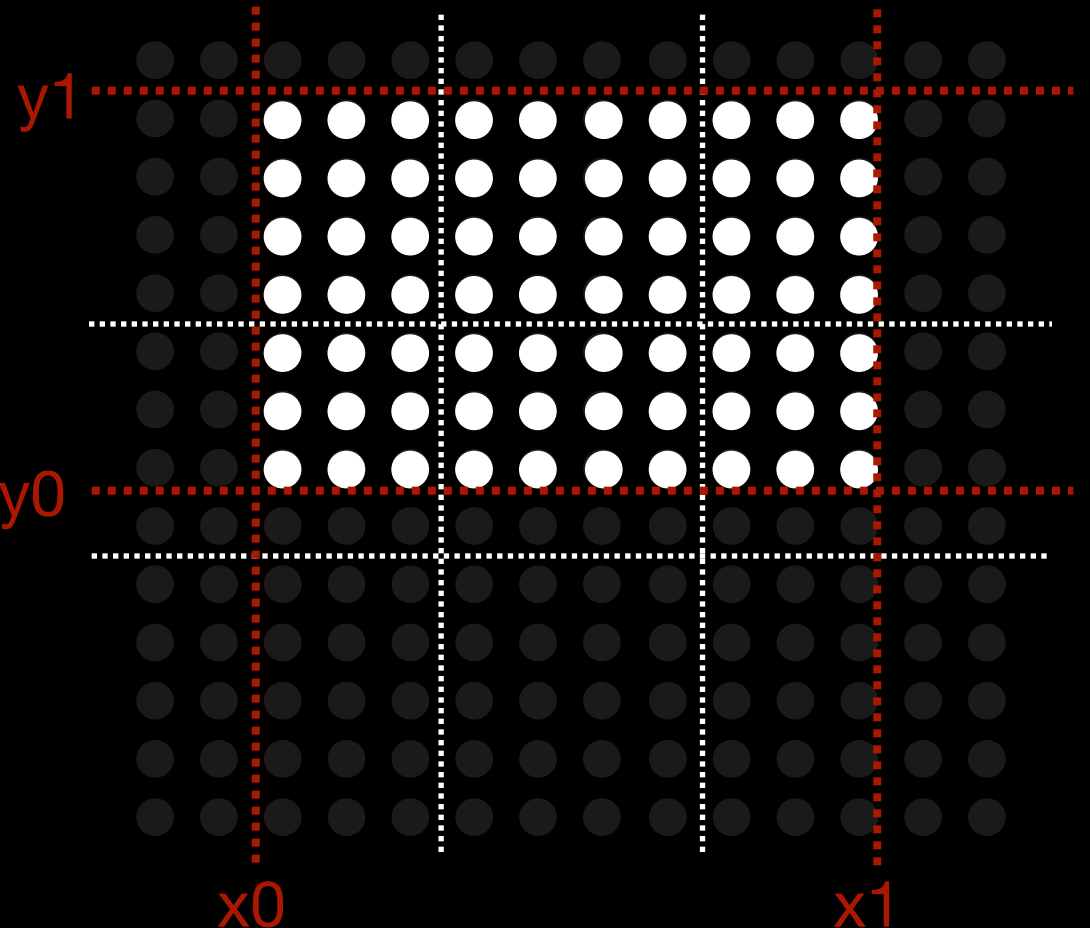


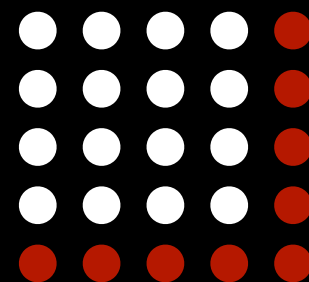
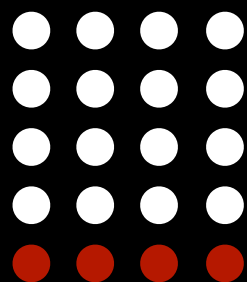
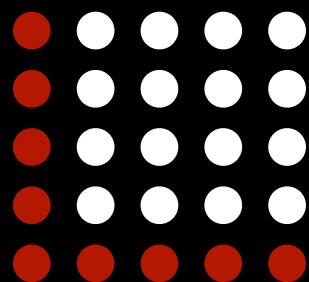
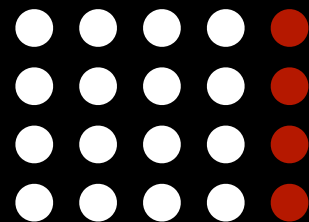
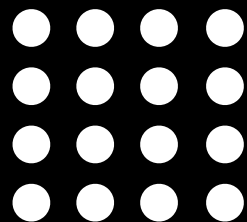
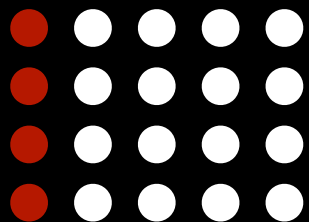
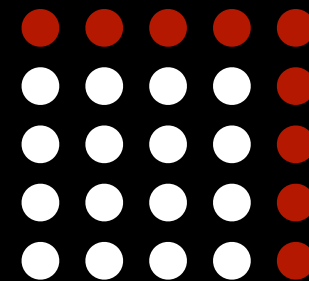
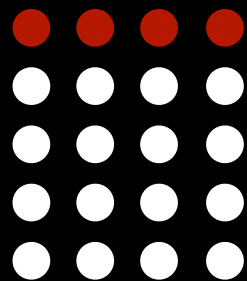
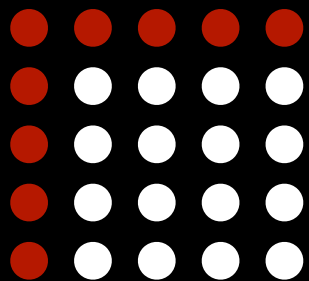
Device spaces.

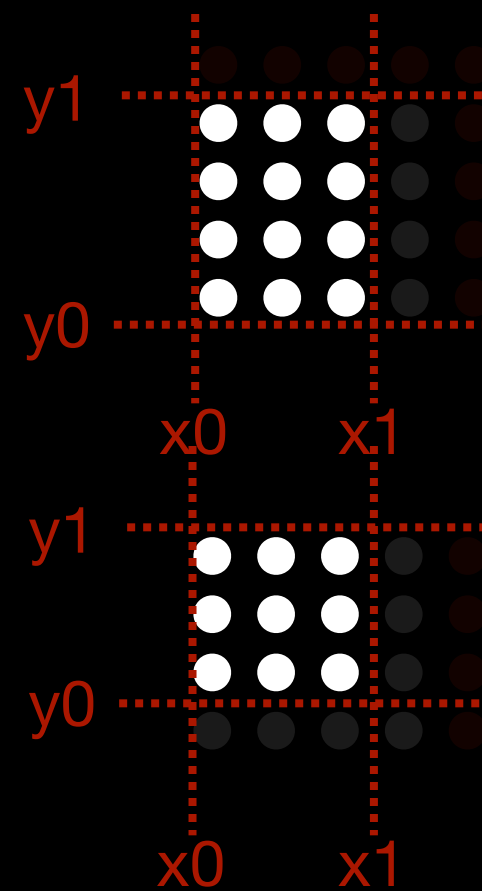
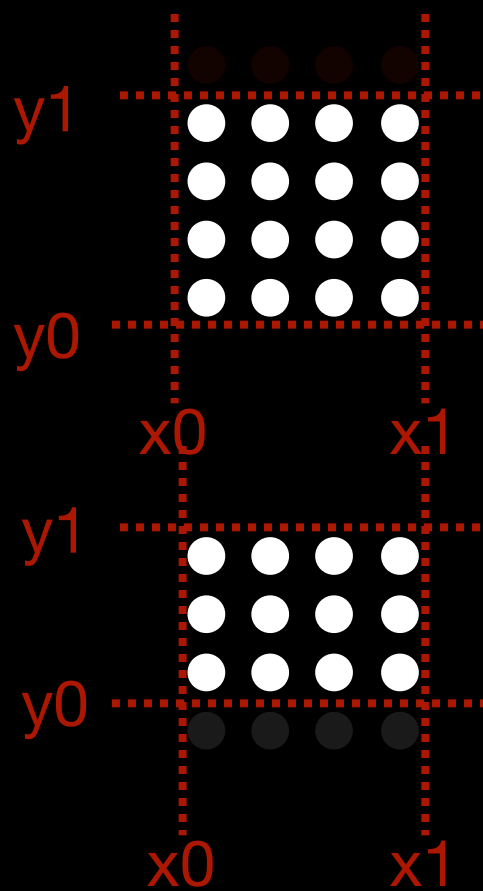
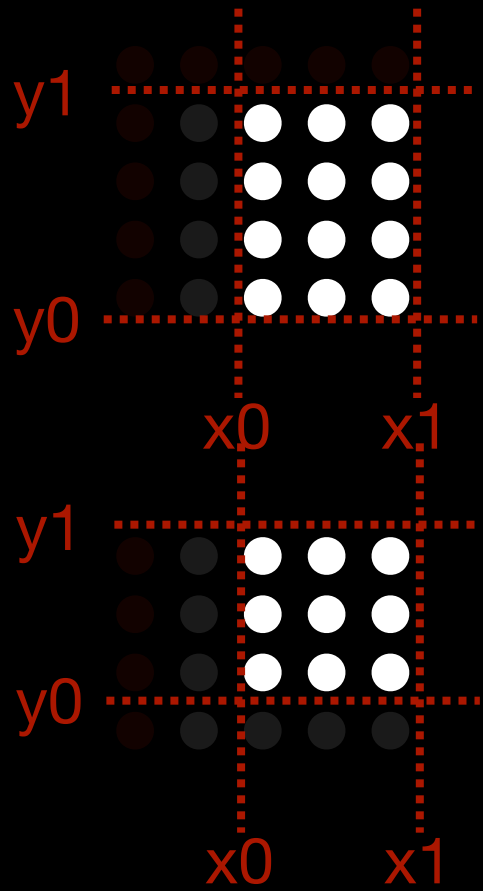
Each device operates within its *space*

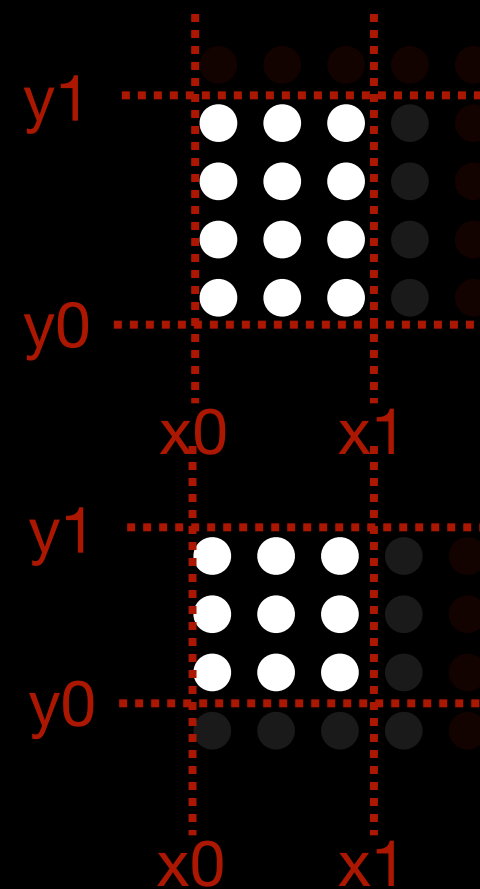
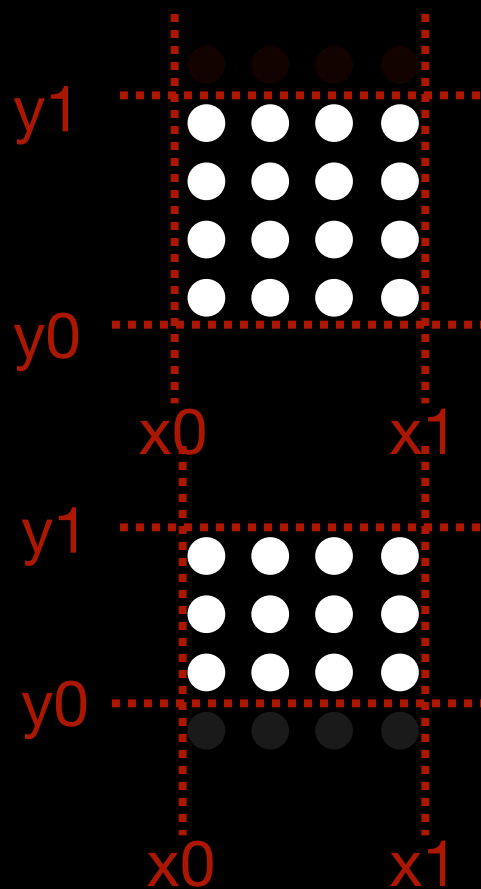
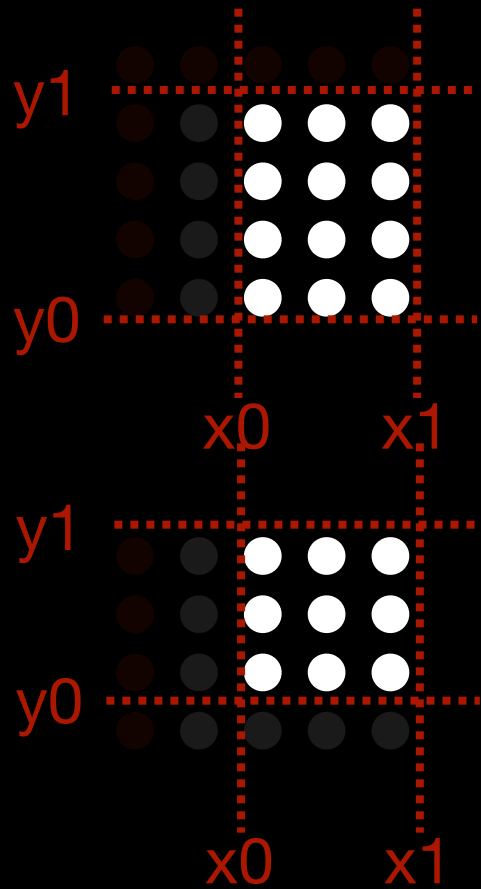
Limits are specified by the user

In parallel, spaces are constrained to their intersection with the local subdomain







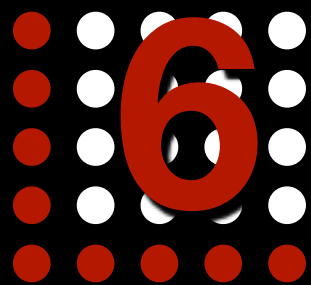
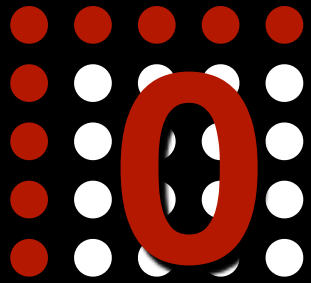


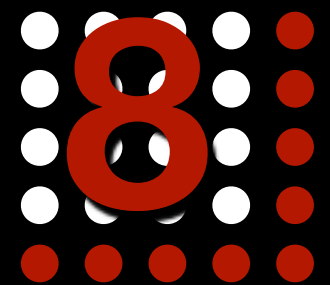
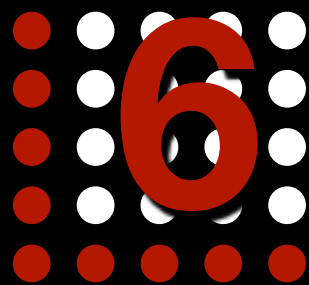
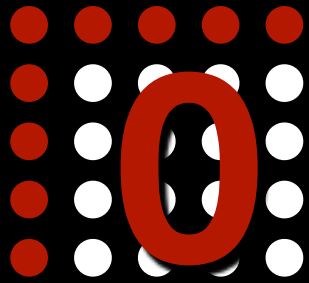
runHere = 0

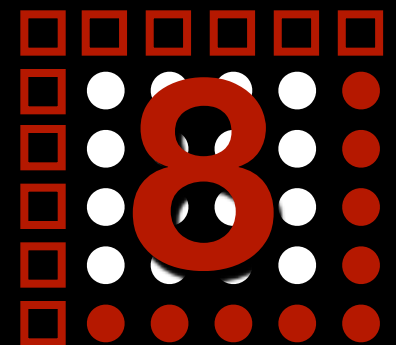
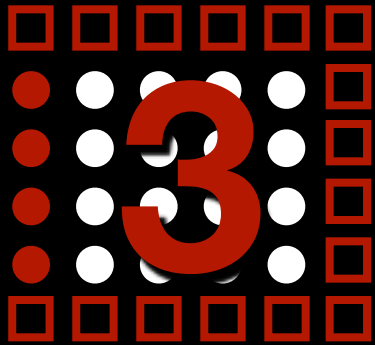
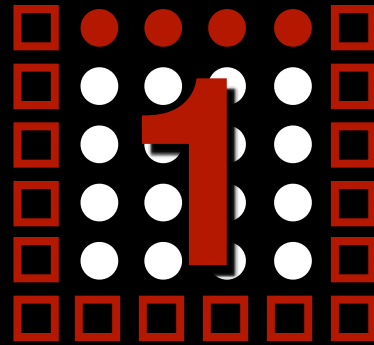
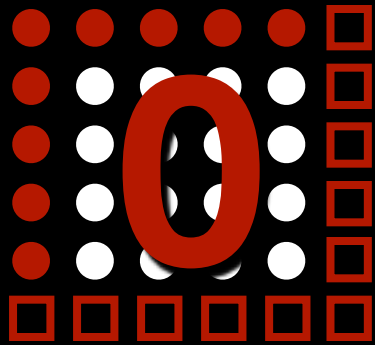
runHere = 0

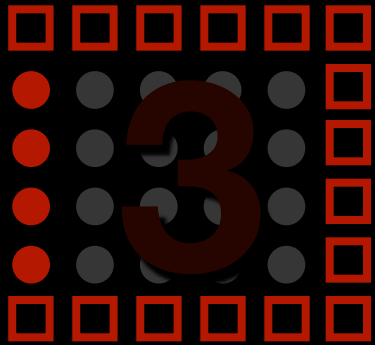
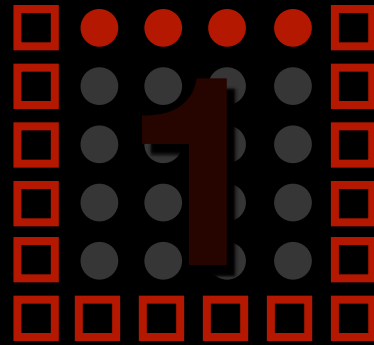
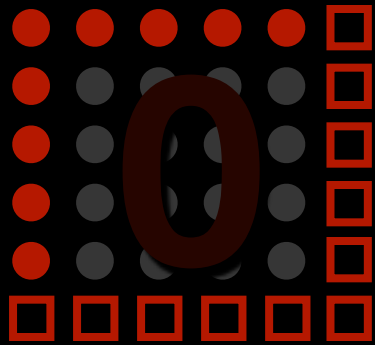
runHere = 0



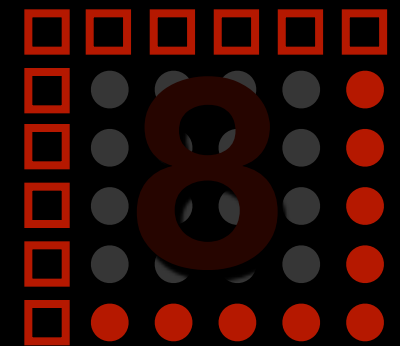
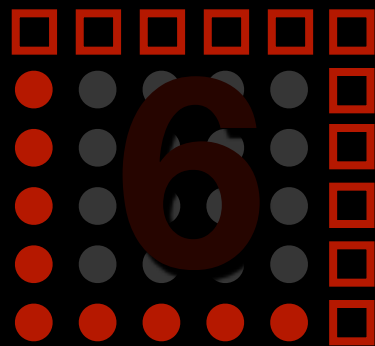
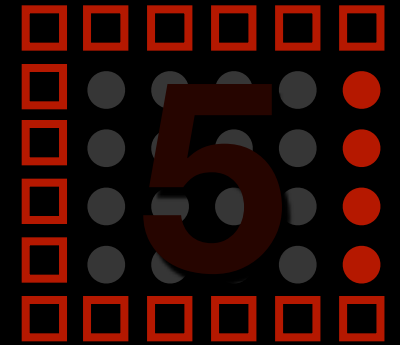


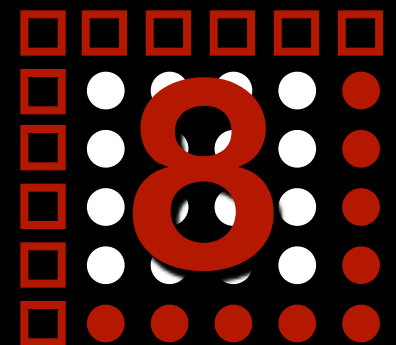
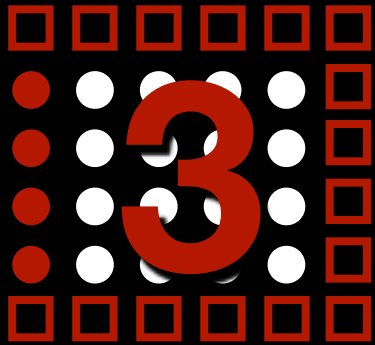
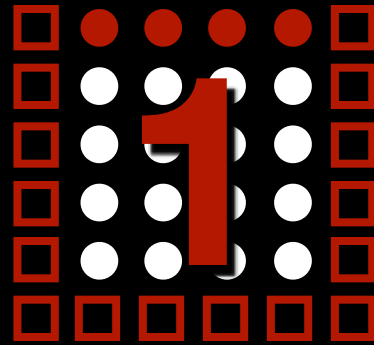
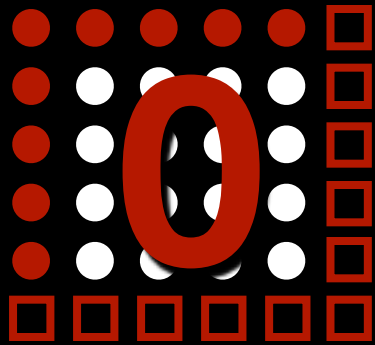


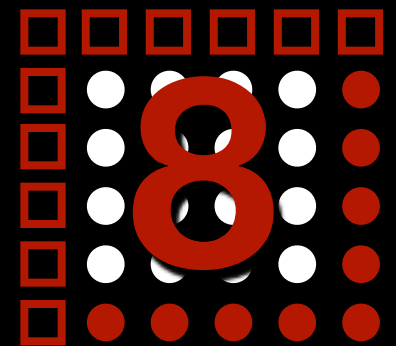
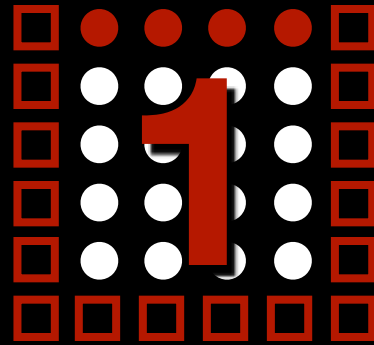
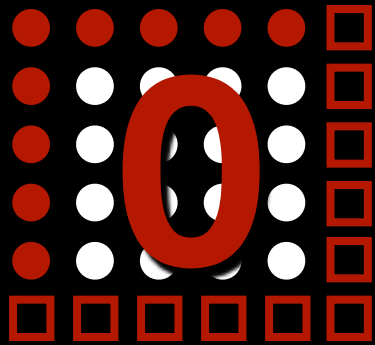


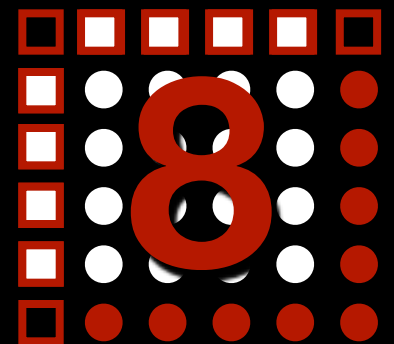
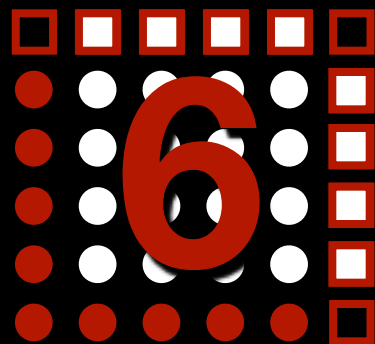
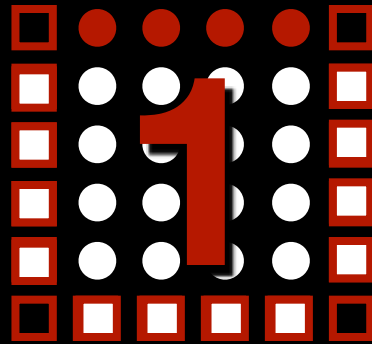
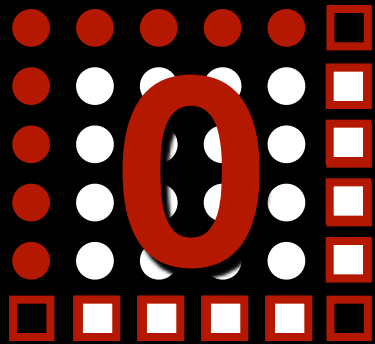


Halo









Communication.



Communication.

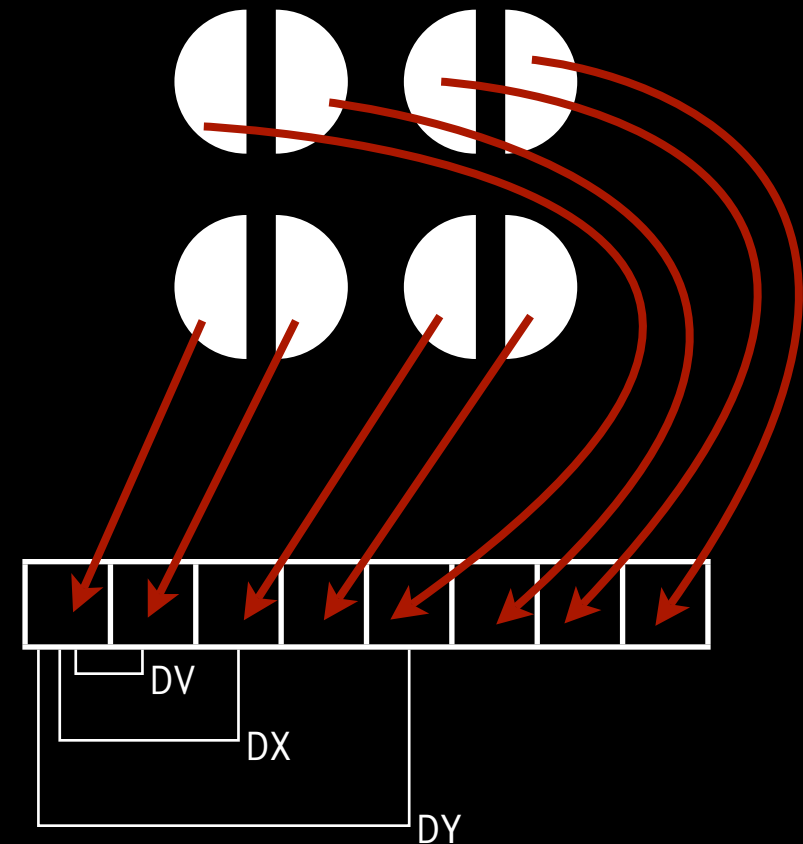
Orthogonal neighbours only



Communication.

Orthogonal neighbours only

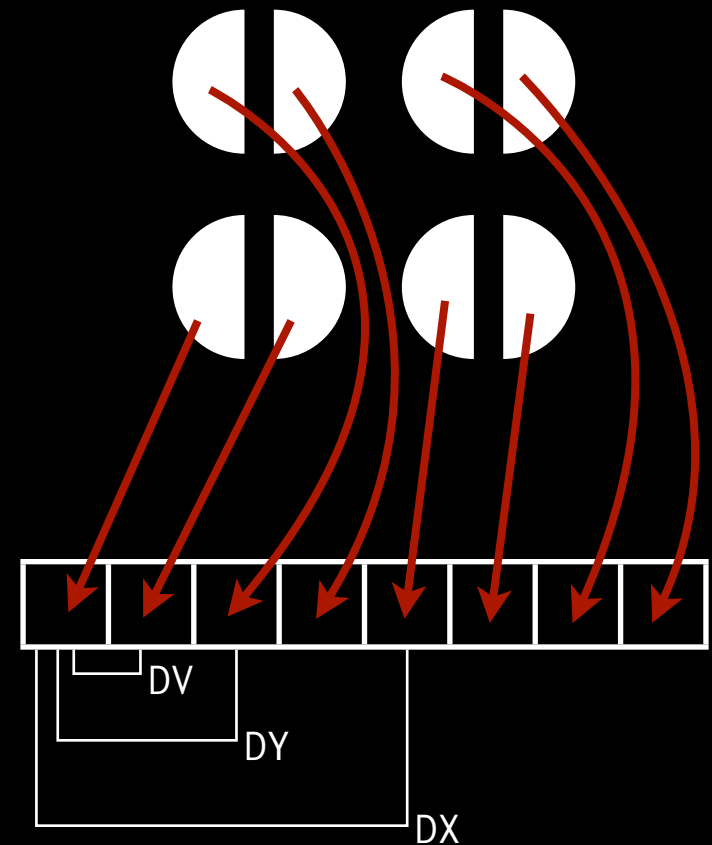
Data being sent must be contiguous in memory or defined as a derived datatype



Communication.

Orthogonal neighbours only

Data being sent must be contiguous in memory or defined as a derived datatype



Optimisation strategies.



Optimisation strategies.

Load Balancing:

Slowest process defines speed of whole program

Favour symmetry



Optimisation strategies.

Load Balancing:

Slowest process defines speed of whole program
Favour symmetry

Minimise Communication:

Communication causes most bottlenecks
Keep subdomains as close to cubic as possible
 $px:py:pz = x_{max}:y_{max}:z_{max}$



C: the cost of communication

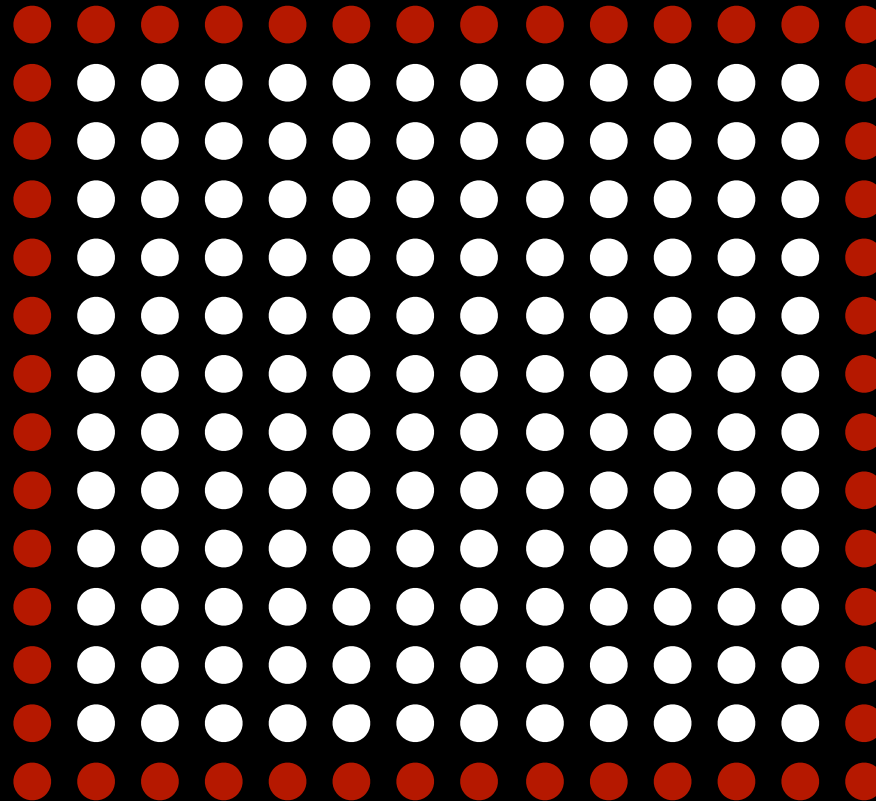
The number of points that must be communicated by a single process during a halo swap:

Worst case—internal processes

Surface area of the subdomain

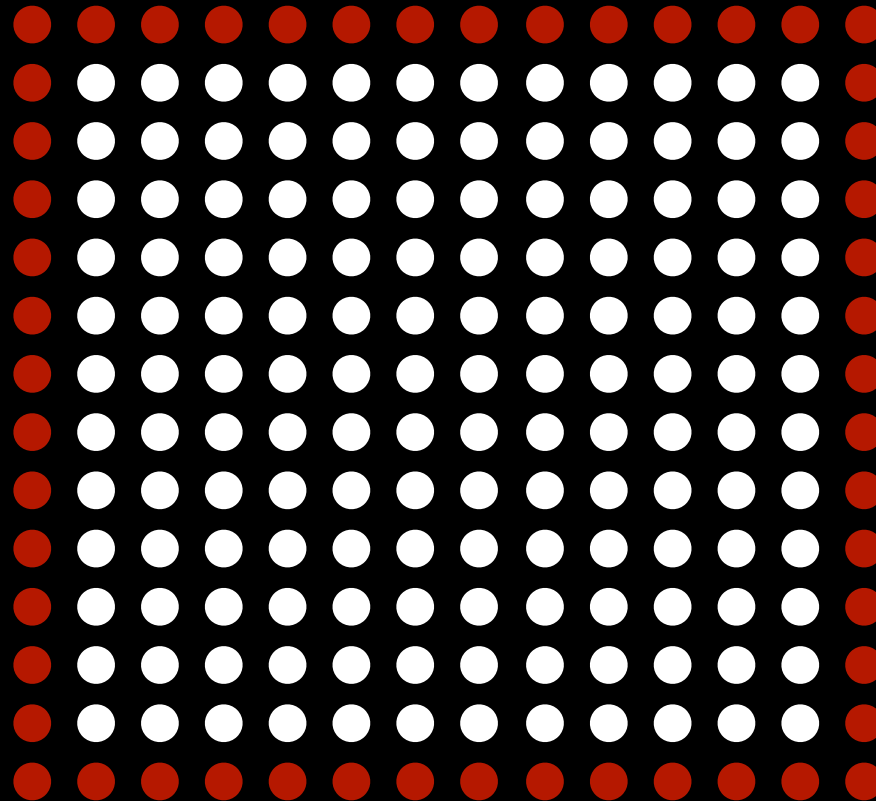


Load balancing.



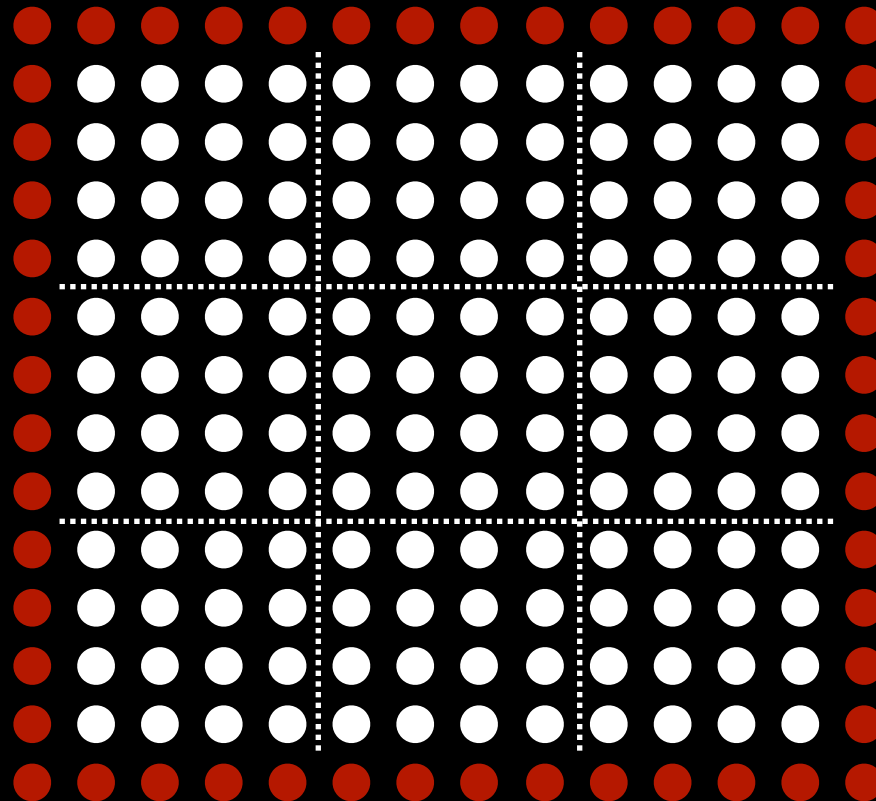
Load balancing.

np = 9



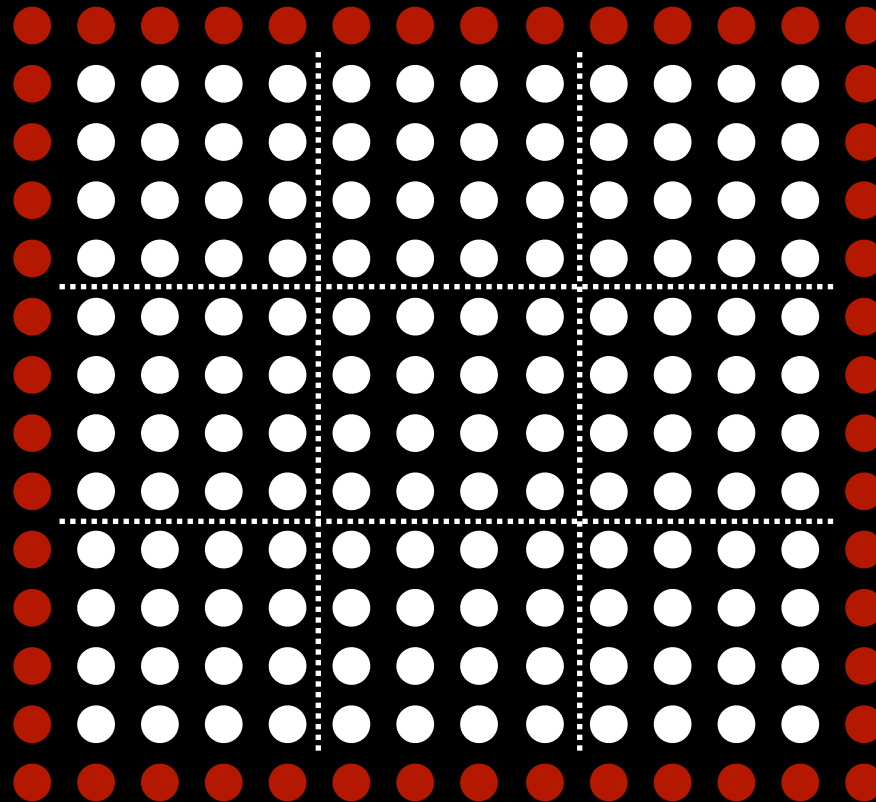
Load balancing.

np = 9



Load balancing.

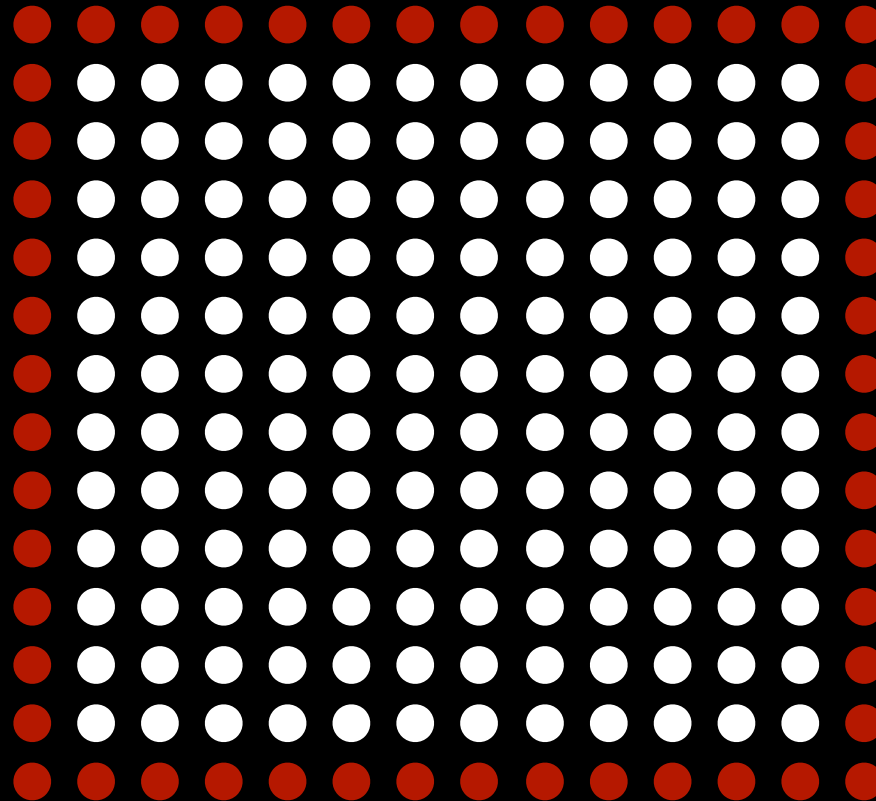
$np = 9$



$C = 16$

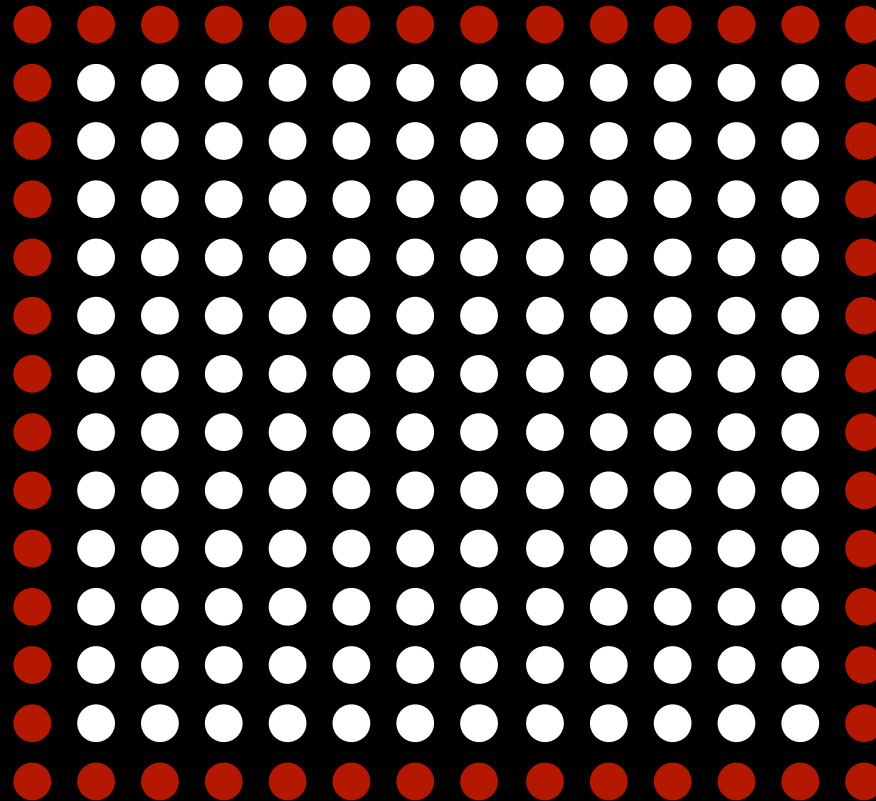


Load balancing.



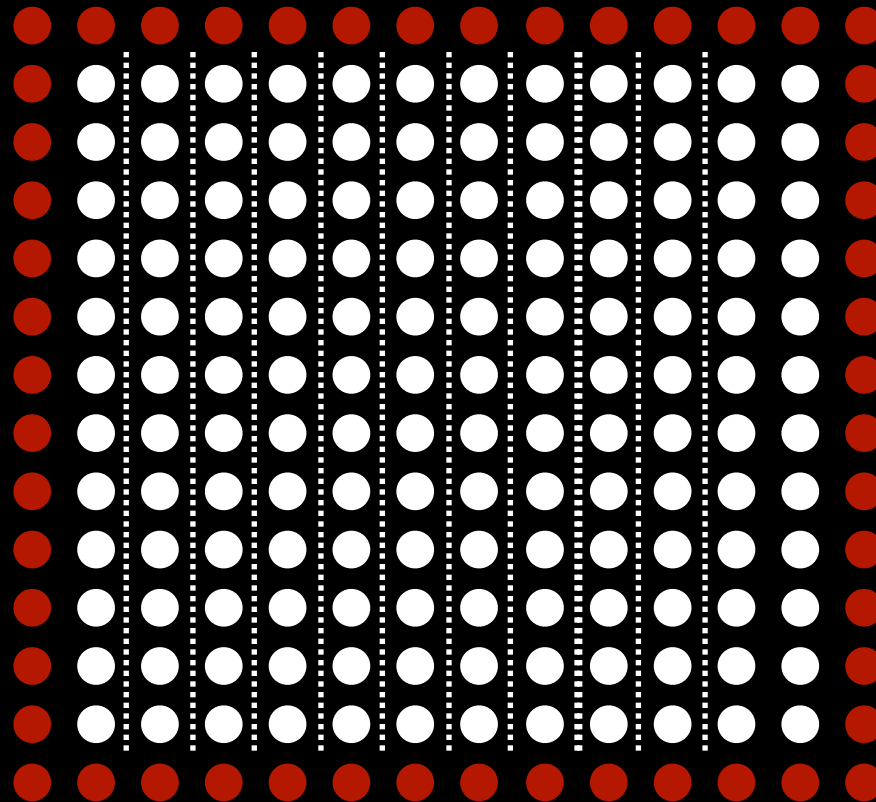
Load balancing.

np = 11



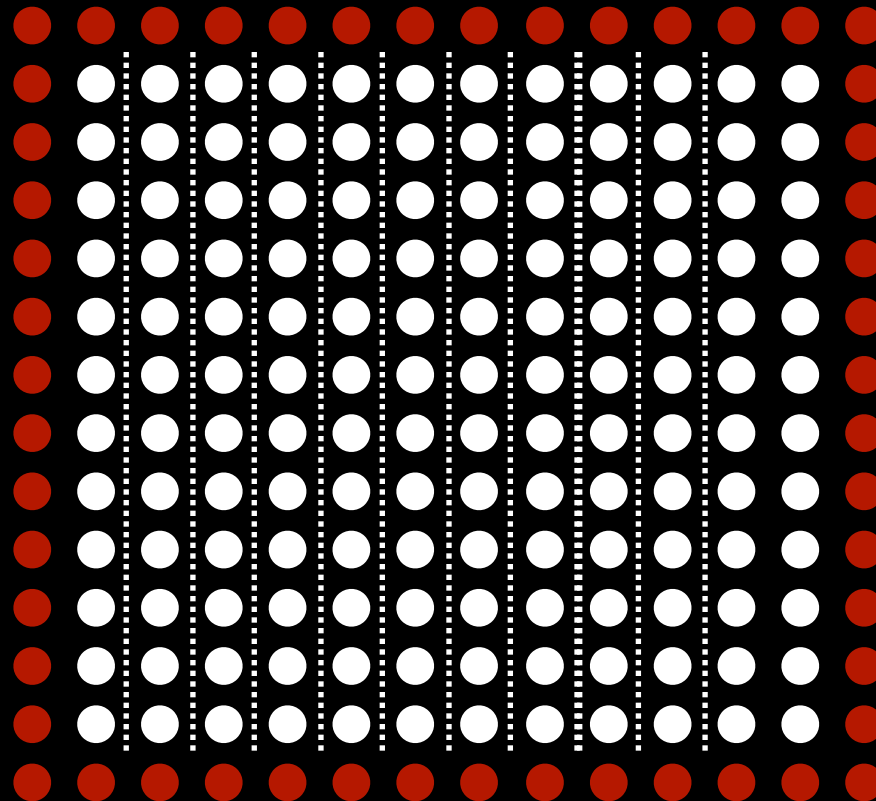
Load balancing.

np = 11



Load balancing.

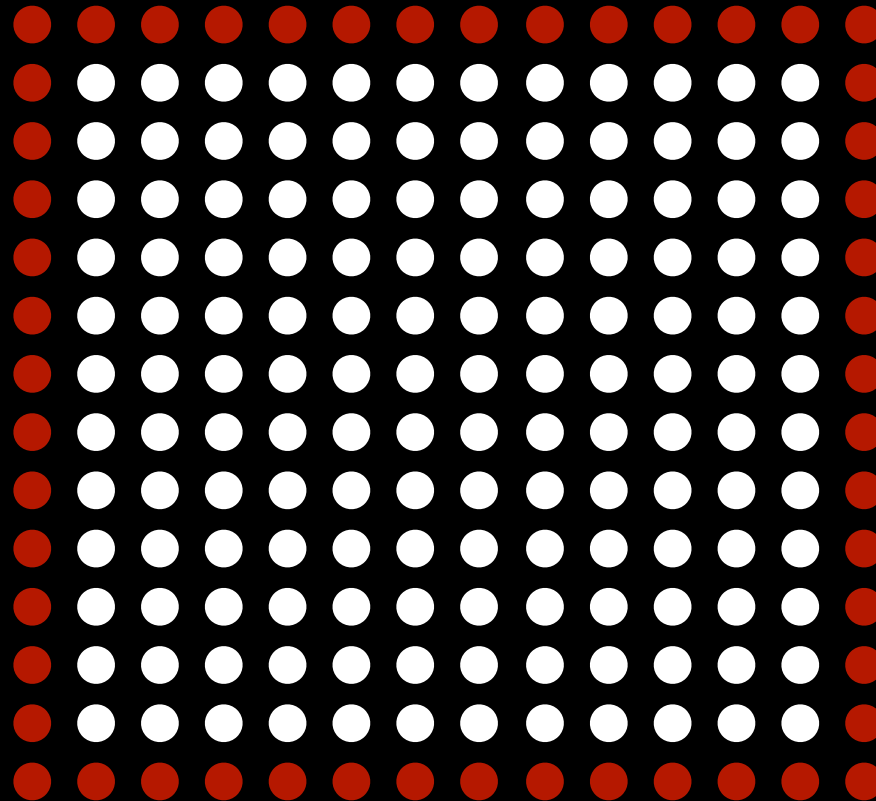
$np = 11$



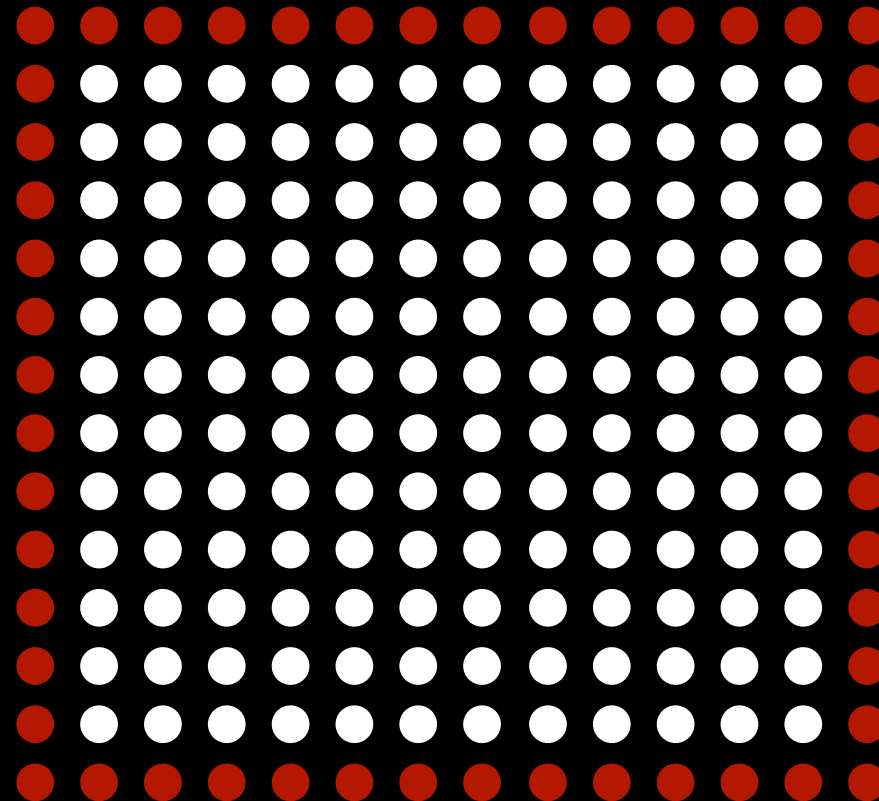
$C = 24$



Minimise communication.



Minimise communication.



$$C = 0$$



Make it *fast!*

Comparing apples with apples.



Comparing apples with apples.

We need to relate supergrid proportions to time.



Comparing apples with apples.

We need to relate supergrid proportions to time.

Computation:

t_P = time to compute 1 point for 1 timestep



Comparing apples with apples.

We need to relate supergrid proportions to time.

Computation:

t_P = time to compute 1 point for 1 timestep

Communication:

t_C = time to communicate 1 point between processes



Comparing apples with apples.

We need to relate supergrid proportions to time.

Computation:

t_P = time to compute 1 point for 1 timestep

Communication:

t_C = time to communicate 1 point between processes

These values must be measured at runtime.



Making it fast.

Minimise t_{TOT} :

$$\begin{aligned}t_{TOT} &= t_P \times vol + t_C \times SA \\ &= t_P \times \left(\frac{x_{max}}{px} \frac{y_{max}}{py} \frac{z_{max}}{pz} \right) + \left(t_C \times 2 \left[\frac{x_{max}}{px} \frac{y_{max}}{py} + \frac{x_{max}}{px} \frac{z_{max}}{pz} + \frac{y_{max}}{py} \frac{z_{max}}{pz} \right] \right)\end{aligned}$$

$$s.t. \quad px \times py \times pz \leq np$$

**Any sufficiently
advanced
technology is
indistinguishable
from **magic**.**

Arthur C. Clarke

Beatbox workflow.

```
state xmax=5 ymax=5 zmax=5 vmax=2;
<std.qui>
<fhn.qui>

def real begin;
def real end;

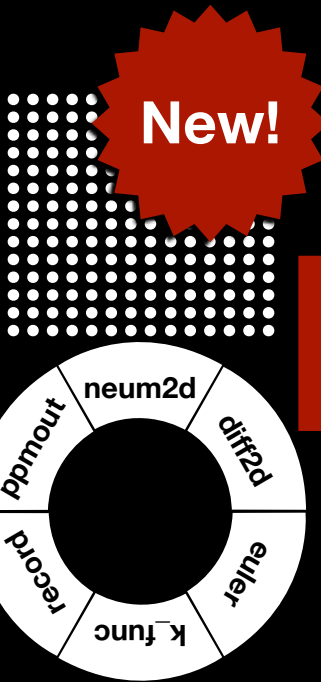
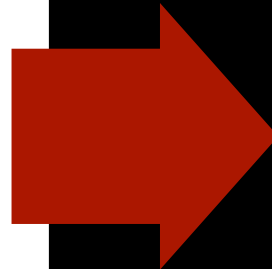
def int printInterval 50;
def real print;

neum2d ;
diff2d v0=0 v1=[IV];
euler v1=2 ode=fhncub par={IV=@2} ;

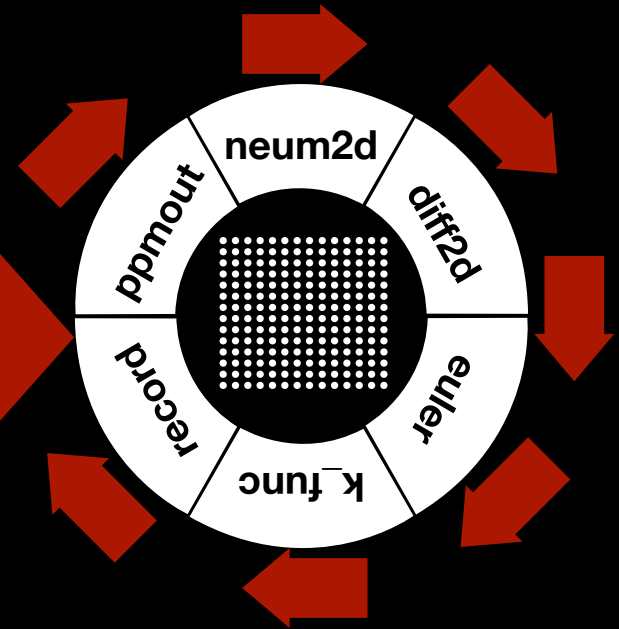
k_func pgm={
  print = eq(mod(t,printInterval),0);
  end = gt(t,10);
};

record when= print file=mySimulation.rec;
ppmout ;
stop when=end;
end;
```

Script



Create



Run

The k_func device.

Allows the user to specify code to be executed as part of the simulation

Creates potential problems for parallelism:

Aggregating values from the whole medium

Further work.

Complete MPI Implementation

Input / Output

Runtime Optimisation

Realistic Geometry

Bidomain Models

Thanks.

Supervisors

Dr. I.V. Biktasheva

Prof. T.B.M Bench-Capon

EPSRC Engineering and Physical Sciences
Research Council