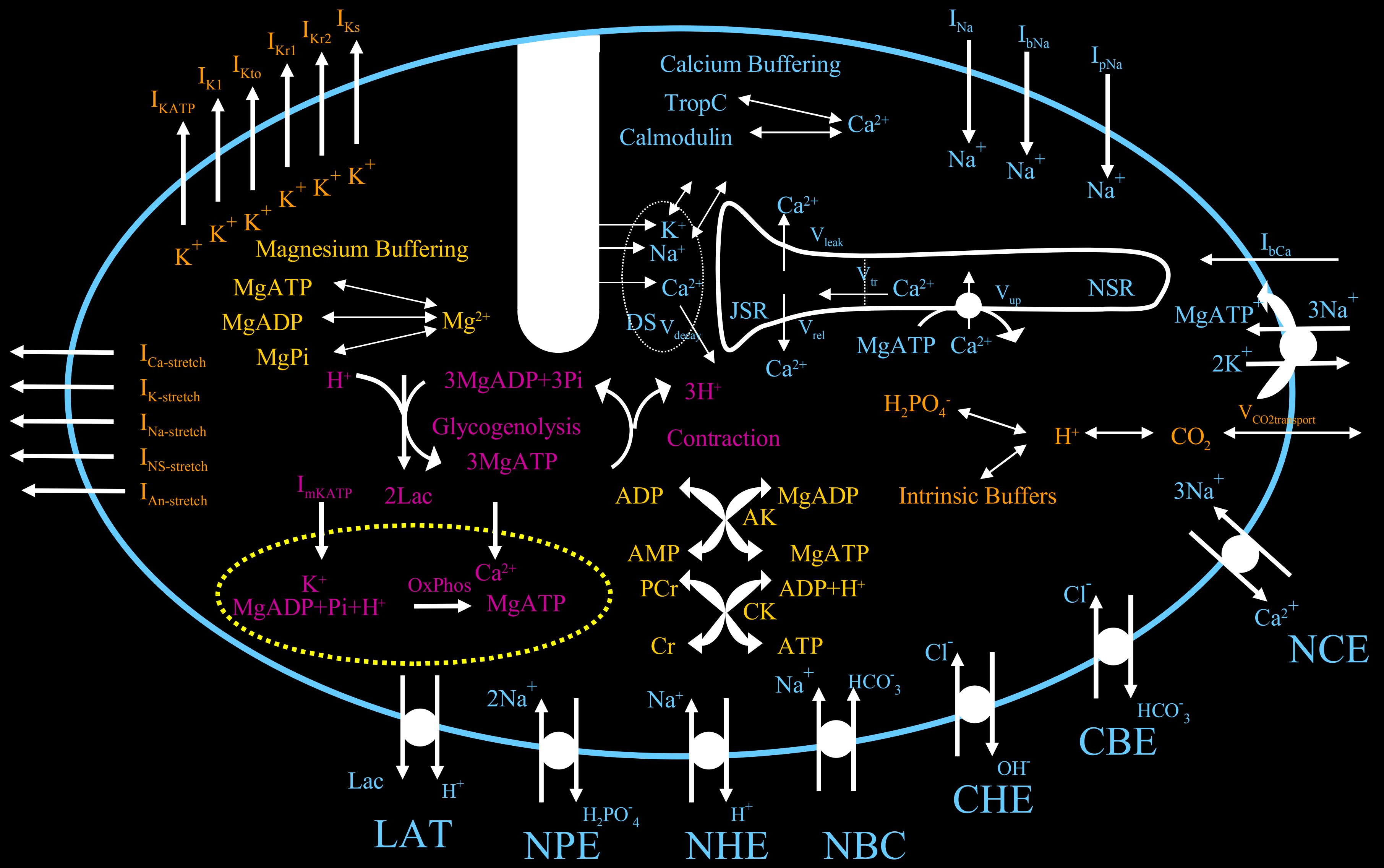
CellML Workshop 2008 – openCMISS and FieldML/CellML

Chris Bradley

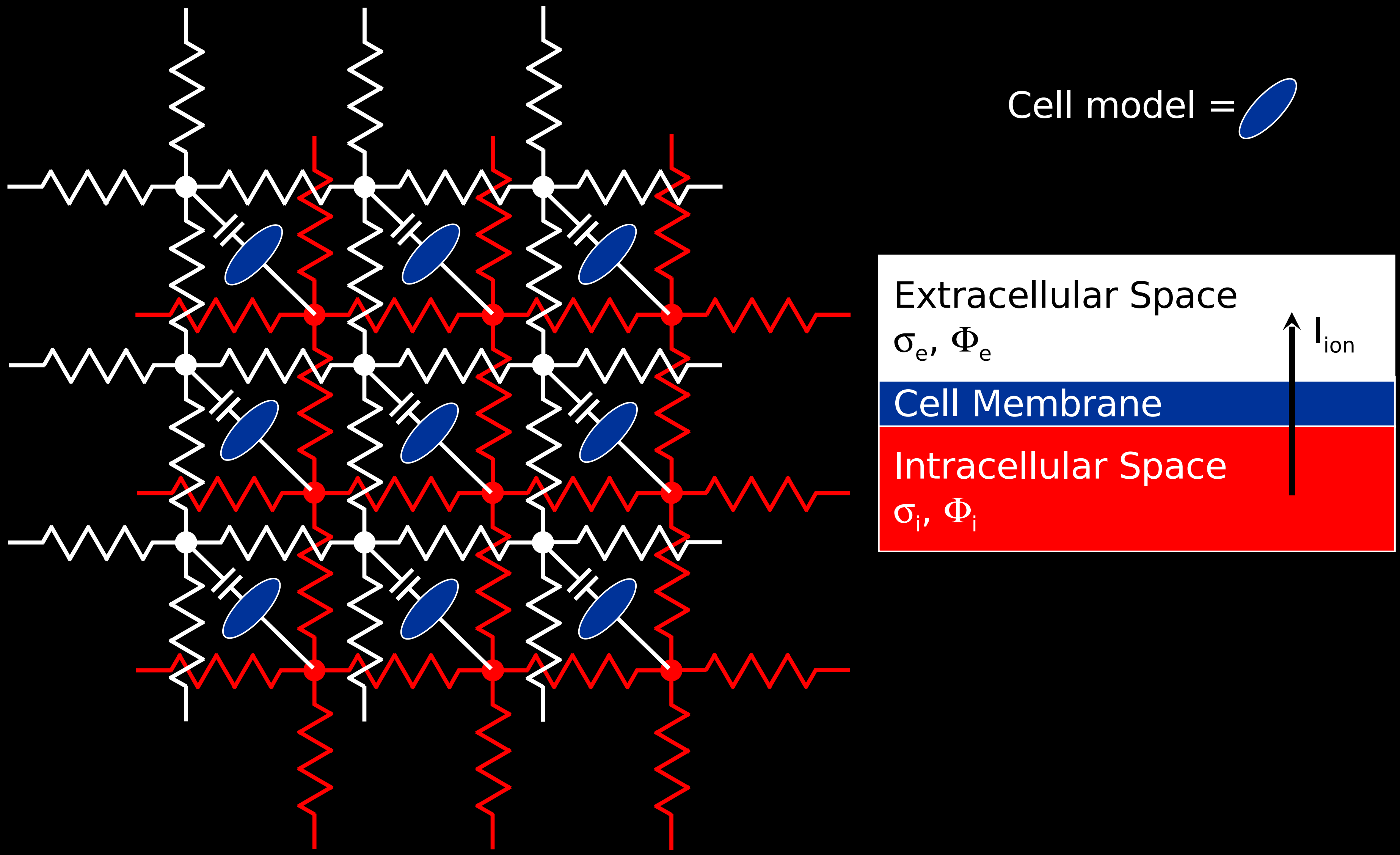
UNIVERSITY OF OXFORD

Department of Physiology, Anatomy and Genetics, University of Oxford

Cellular processes



Bidomain representation



Bidomain equations

$$\nabla\cdot[(\sigma_i+\sigma_e)\nabla\Phi_e] = -\nabla\cdot(\sigma_i\nabla V_m)+I_{stim1}$$

$$\nabla\cdot(\sigma_i\nabla V_m)+\nabla\cdot(\sigma_e\nabla\Phi_e) = A_m(C_m\frac{\partial V_m}{\partial t} + I_{ion})-I_{stim2}$$

$$V_m = \Phi_e - \Phi_i$$

Where:

- V_m = transmembrane potential
- C_m = membrane capacitance
- A_m = membrane surface/volume ratio
- I_{ion} = ionic source current from cellular model
- I_{stim} = stimulus current

  Φ_e = extracellular potential

  Φ_i = intracellular potential

  $\boldsymbol{\sigma}_e$ = intracellular conductivity tensor

  $\boldsymbol{\sigma}_i$ = intracellular conductivity tensor

BOTE problem size calculation

- Future electrical activation problem:
- Average human heart is approx 130 mm x 90 mm x 70 mm = 8.19×10^5 mm^3. Assume 50% is ventricle.
- For 100 μm spacing would need 4.23×10^8 computational points (cp).
- For 30 ODEs/cp we would have 1.27×10^{10} ODEs to solve at each time instance.
- Assuming 100 flop/ODE we would have 1.27×10^{12} flop per time instance.
- A 1 ms time step will give 1.27×10^{15} flop/s real simulation time or 5.26×10^{16} flop/min real time.

Solution time calculation

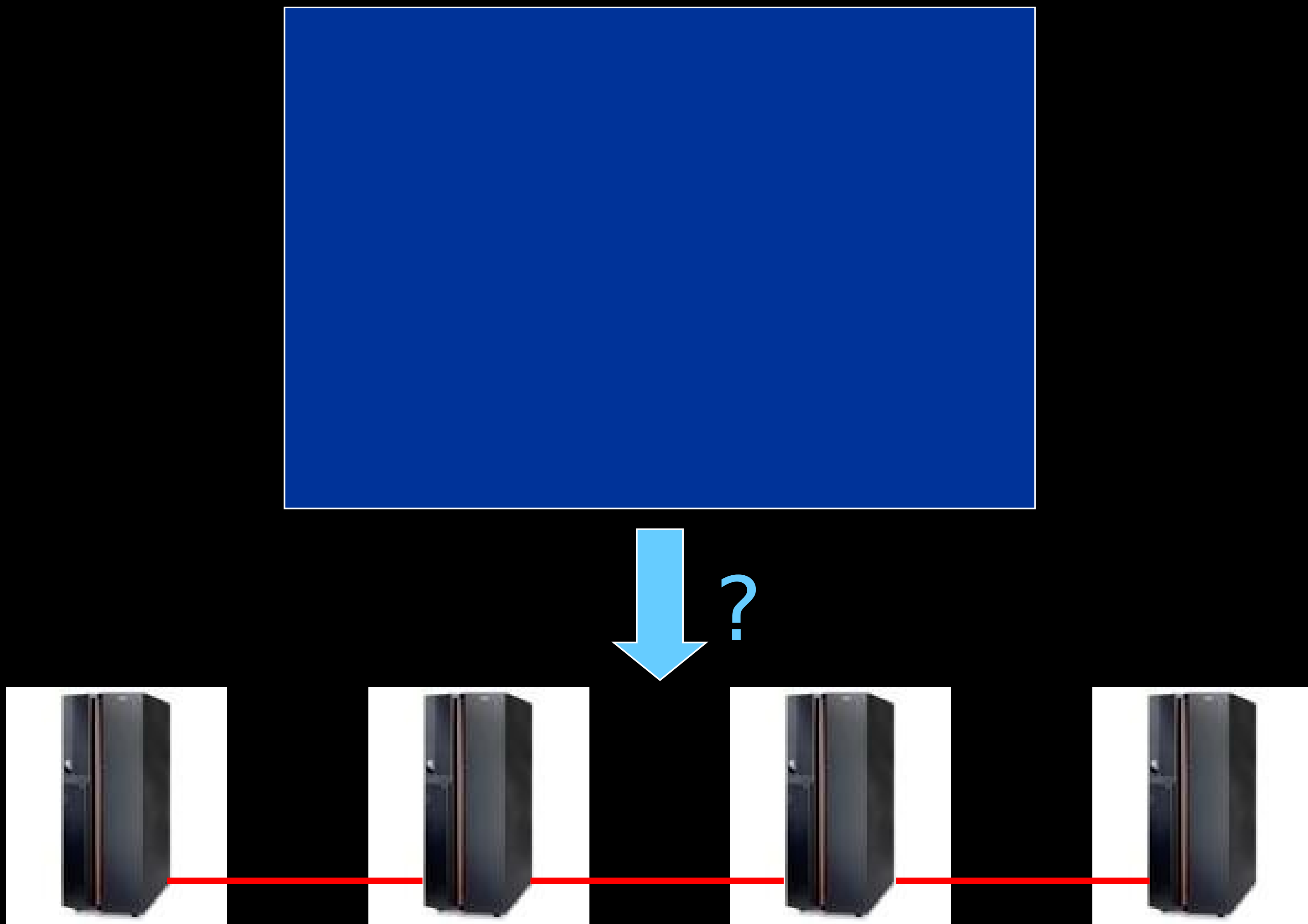
- A 2.4 GHz Intel Core 2 can achieve ~ 1.7 Gflops with linpack (best case!)
- To simulate 1 min real time on 1 processor would therefore take 4.48×10^7 s or 518.5 days!
- Or, in other words, to get the solution time down to 1 day would require 519 processors assuming perfect speedup (not very likely!)

- We are dealing with very, very big problems that require large parallel computers!

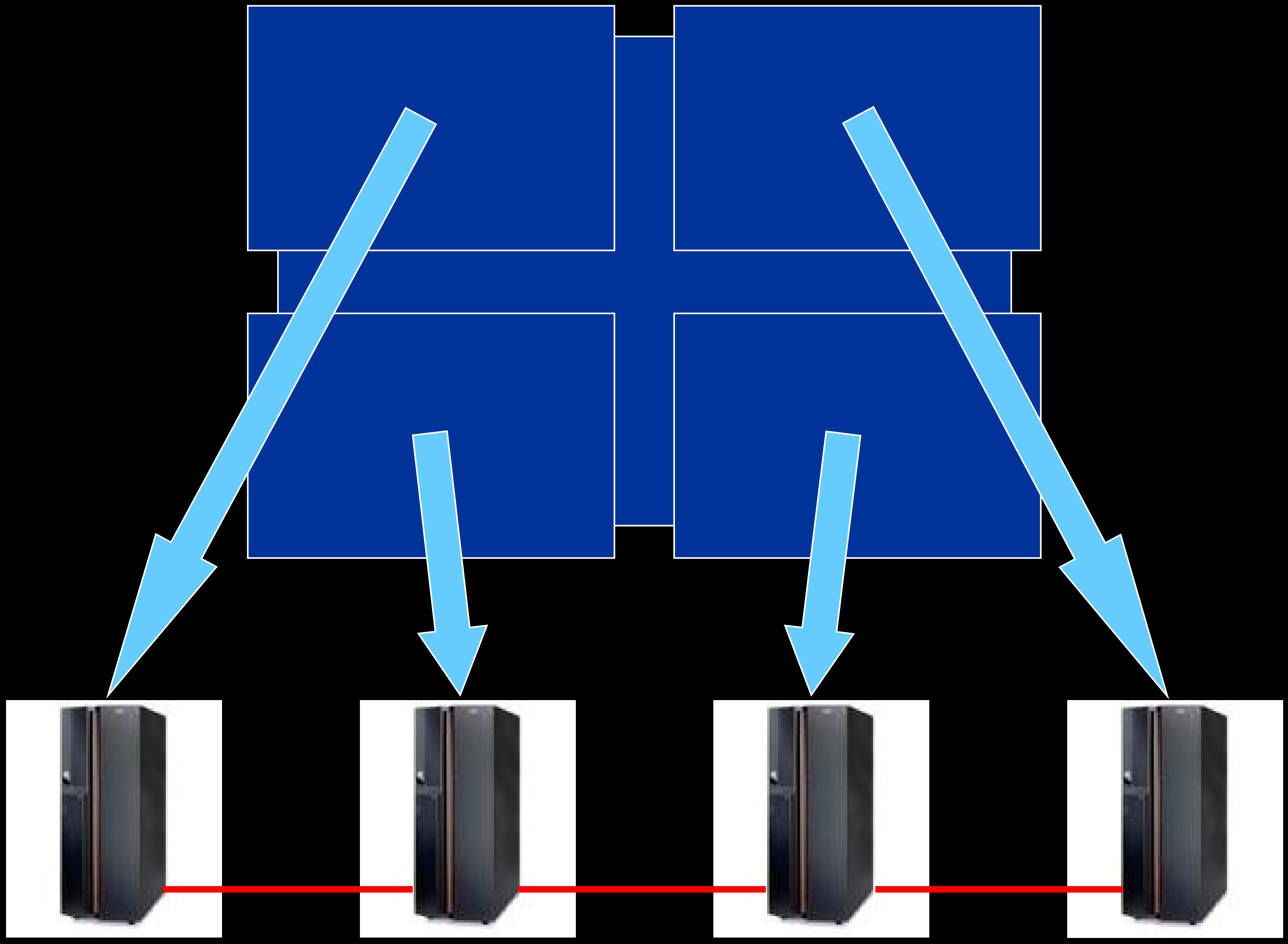
openCMISS

- Re-engineering of CMISS computational engine.
- Want
  - a library based approach to enable use in multiple applications.
  - Modular, easily extendable and programmable.
  - MPI based distributed memory + OpenMP + serial code
  - Easy to understand and program by novices.
- Open source on sourceforge
  - http://sourceforge.net/projects/opencmiss
- Fortran 95.
- Object “based”
- Still in the preliminary stage – about 73,000 lines.

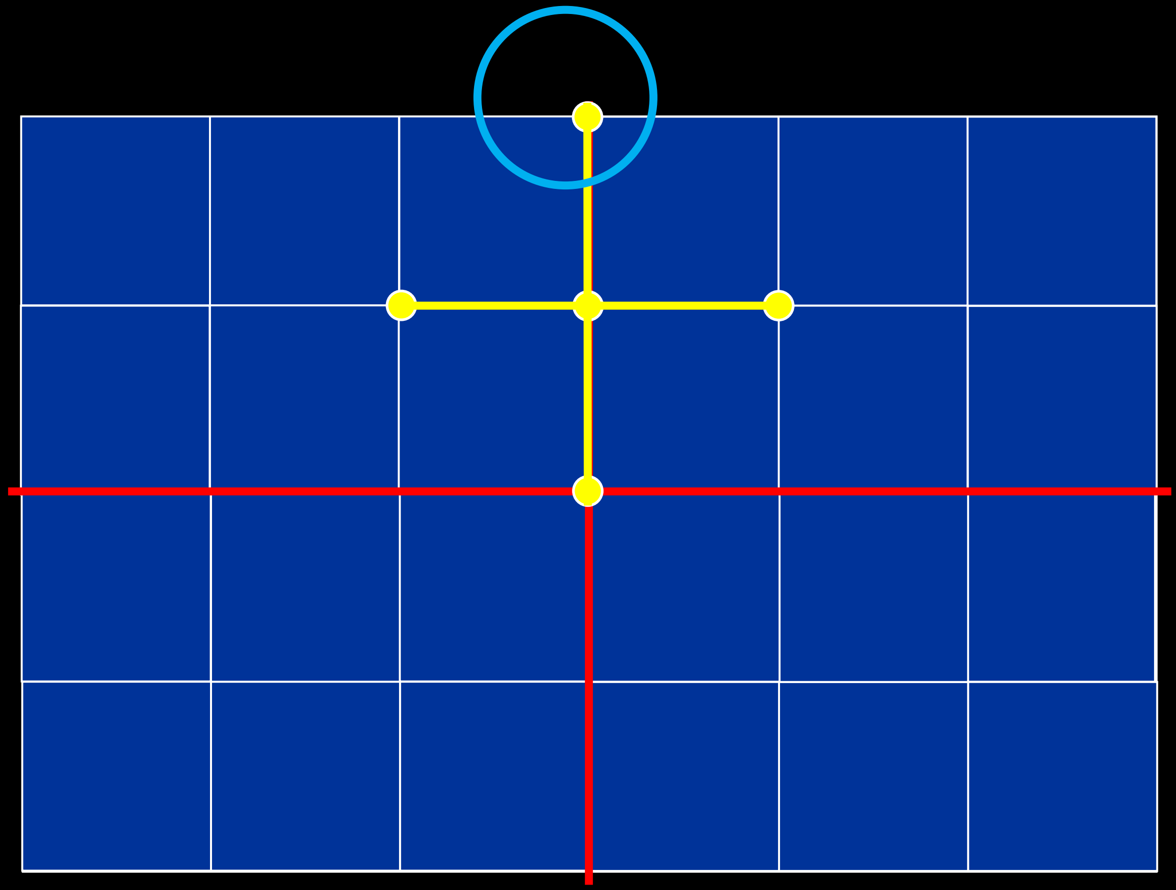
Parallelisation?



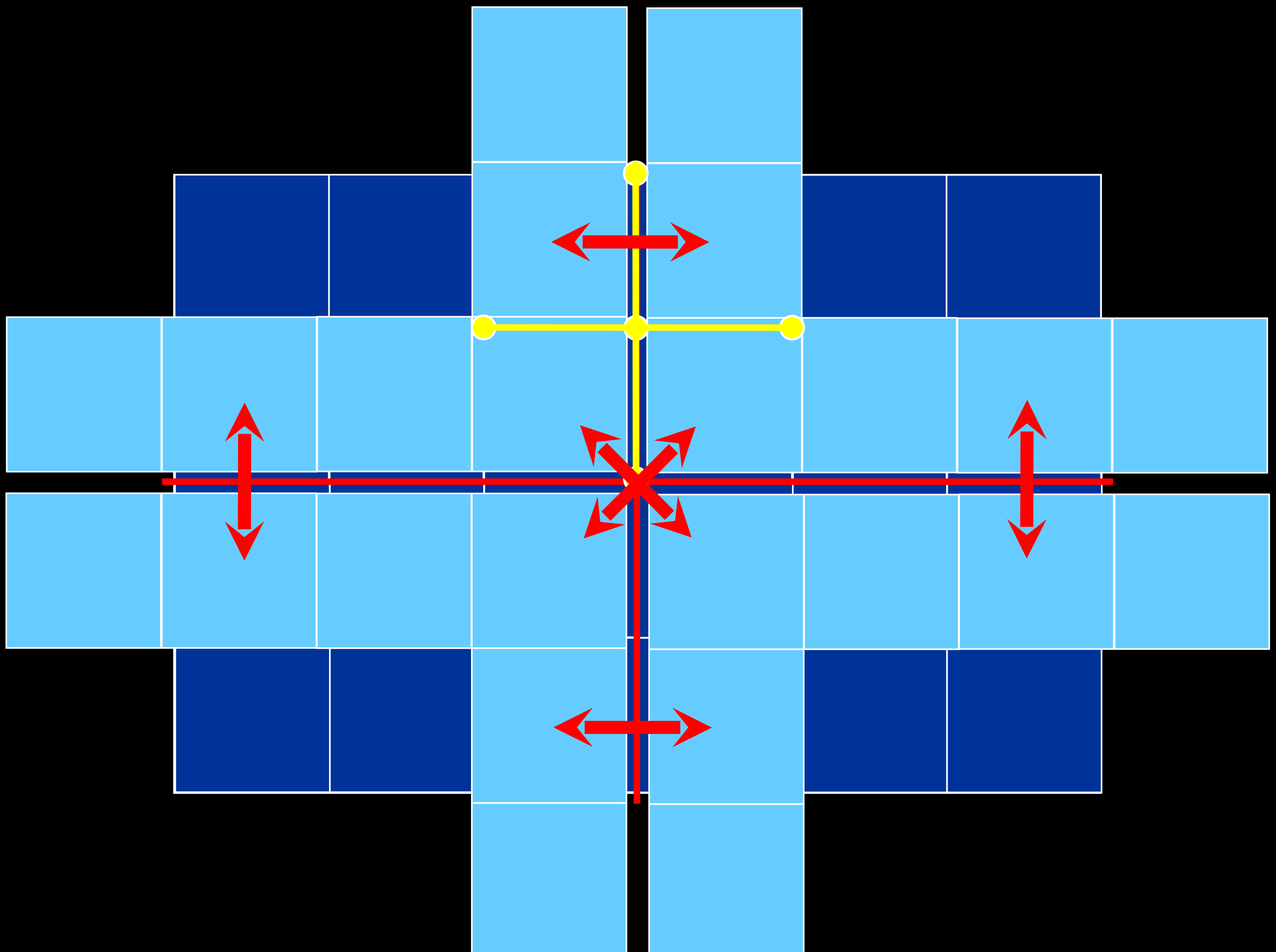
Domain decomposition



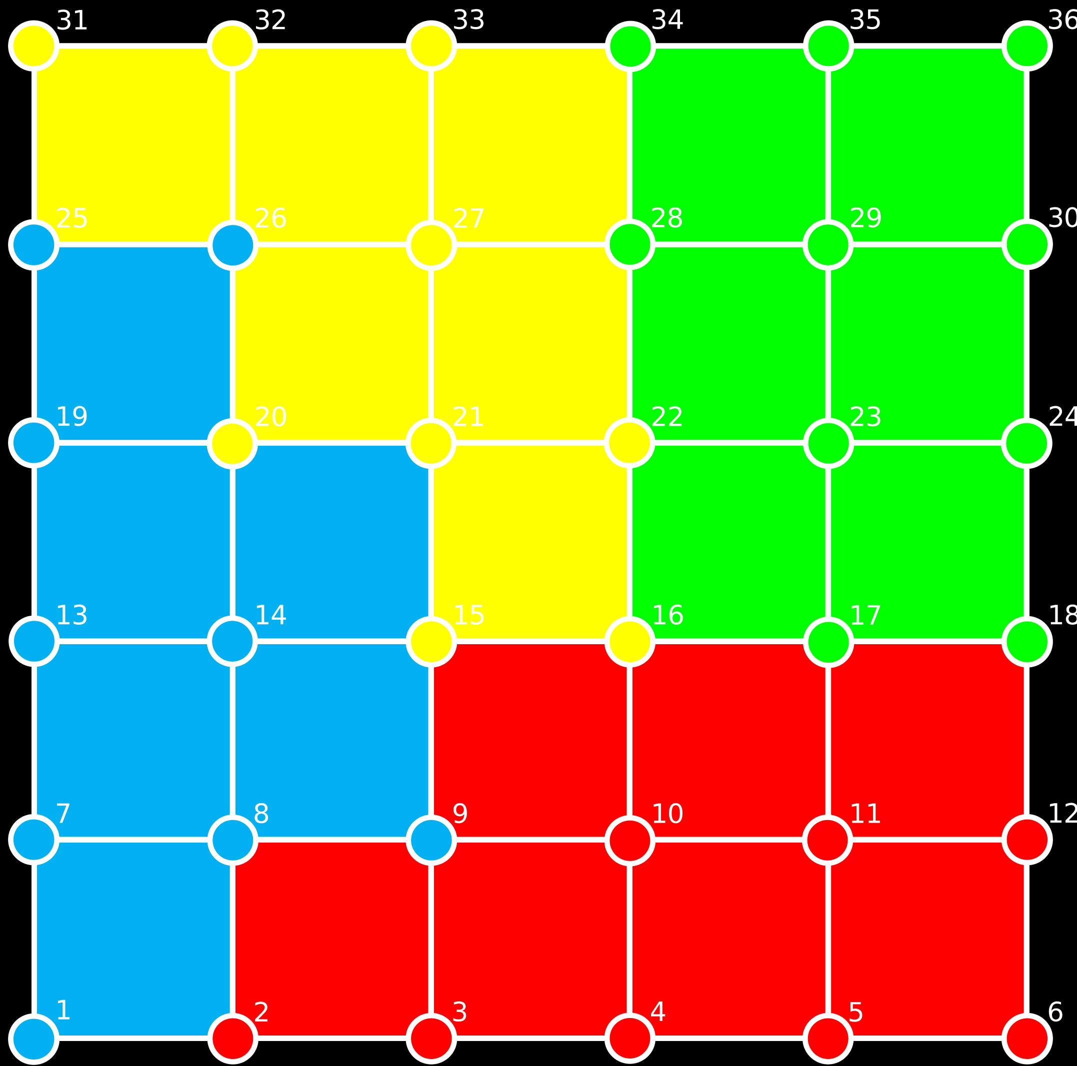
Domain decomposition



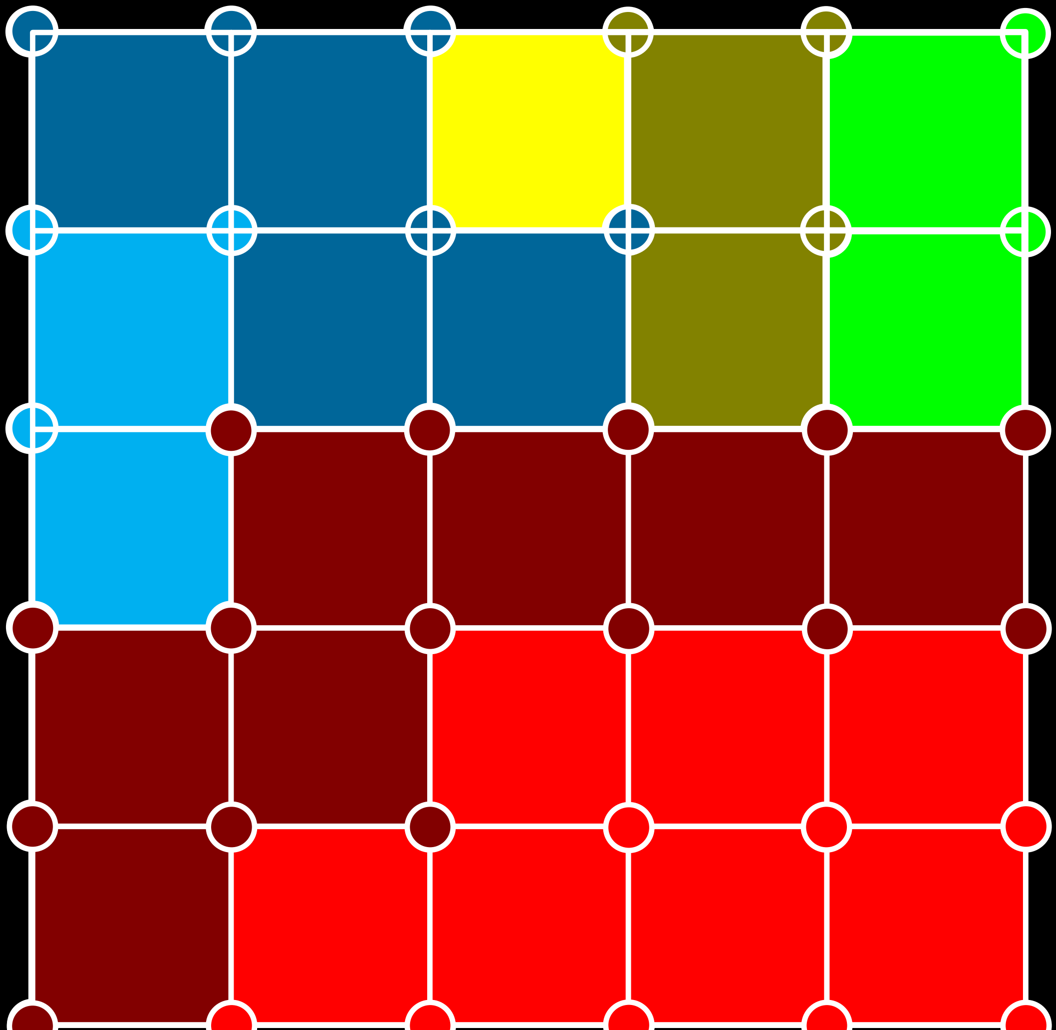
Ghosting



Mesh Decomposition



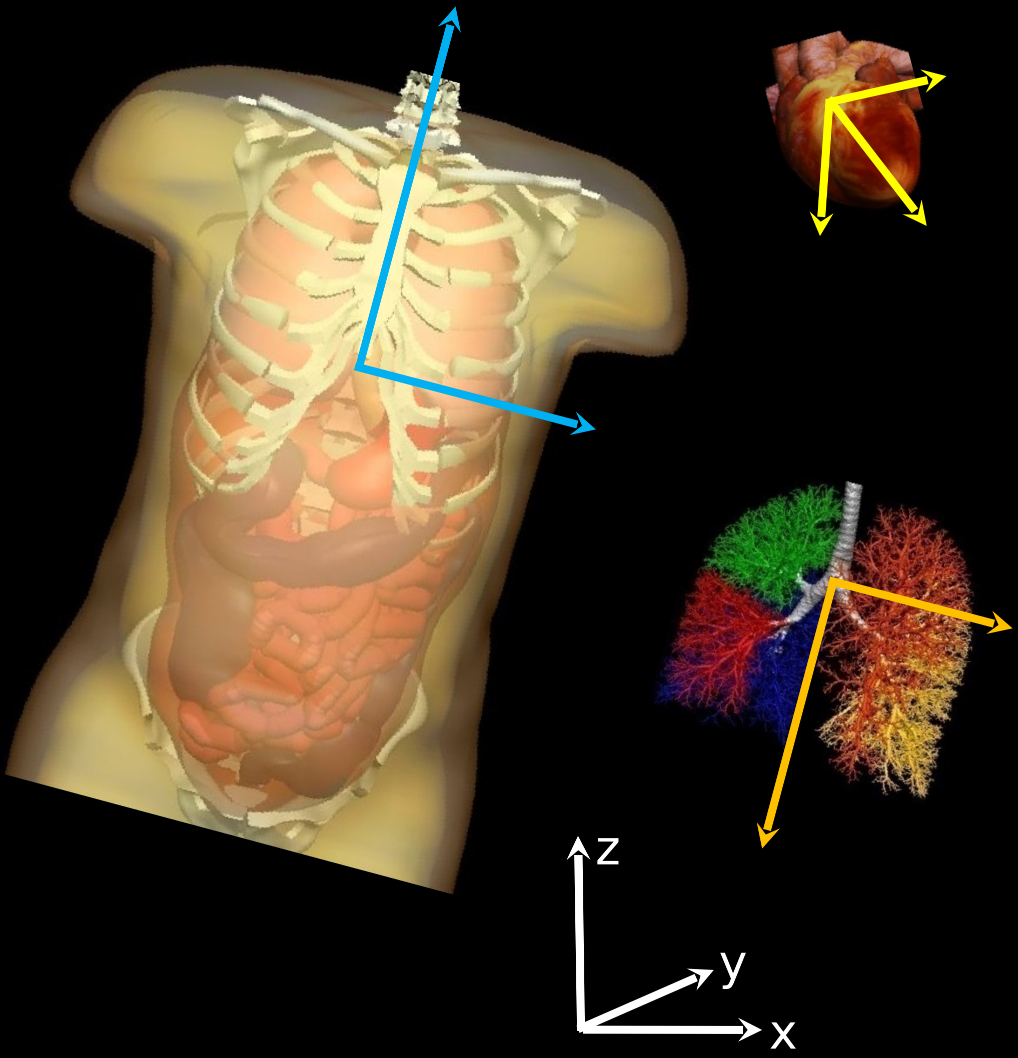
Mesh Decomposition



Regions

Regions contain:

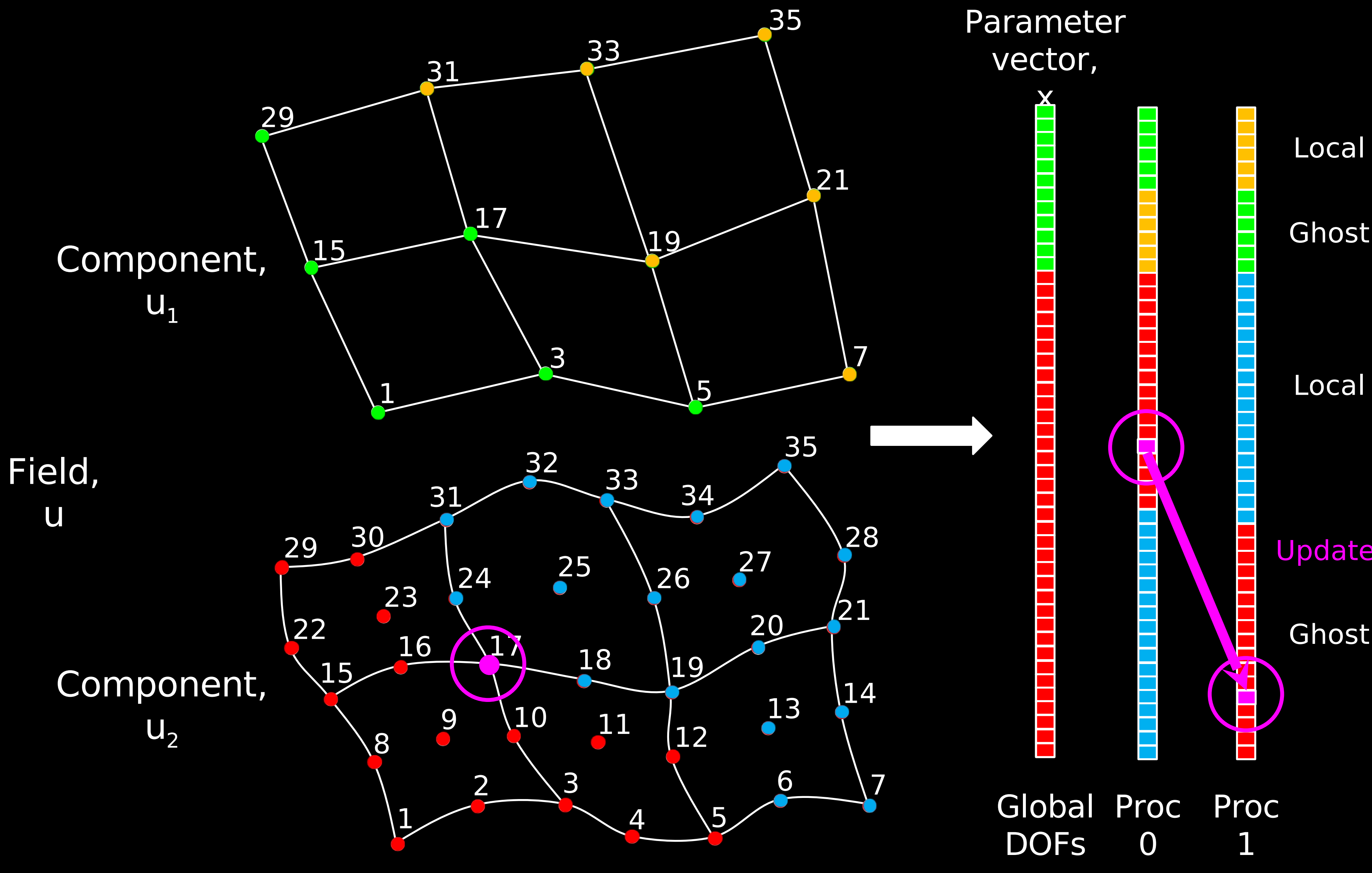
- Coordinate system
- Nodes
- Meshes
- Fields
- Problems
- Daughter and parent regions



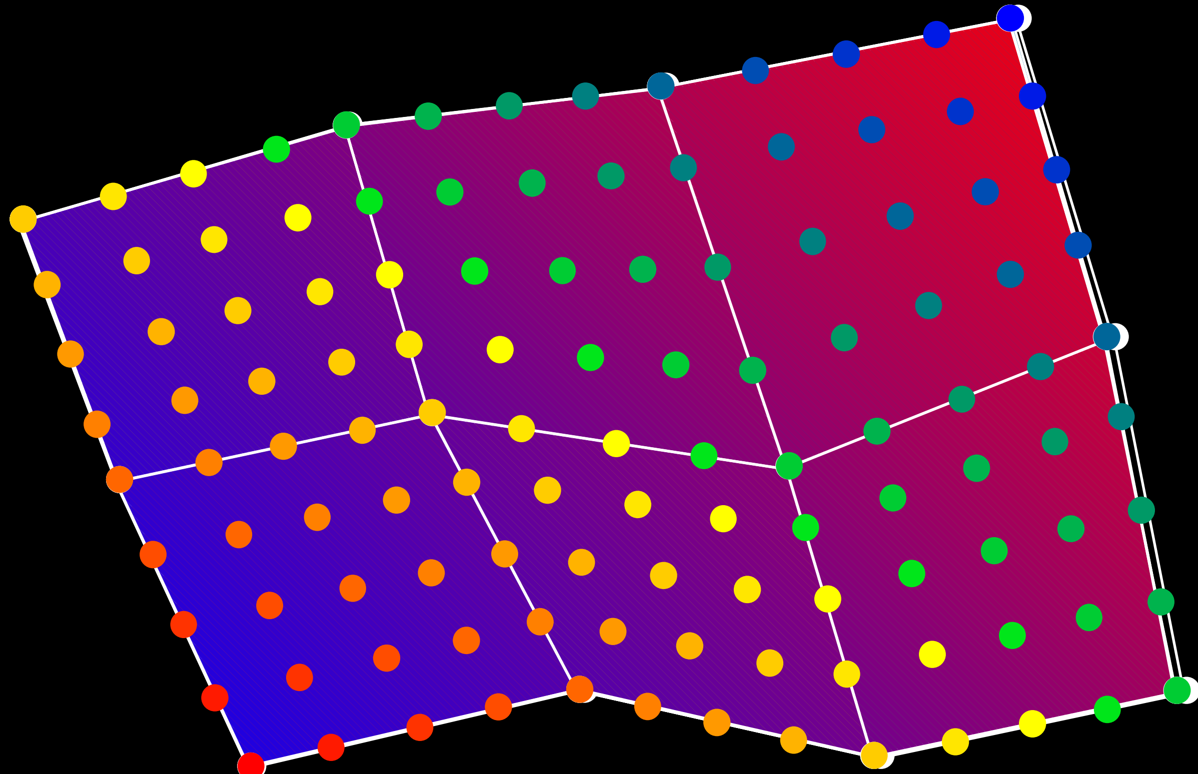
Fields

- Fields are the central object for storing information and framing the problem.
- Fields have a number of field variables i.e., u, $\partial u/\partial n$, $\partial u/\partial t$, $\partial^2 u/\partial t^2$.
- Each field variable has a number of components.
- A field is defined on a decomposed (broken up) mesh.

Field parameter vector



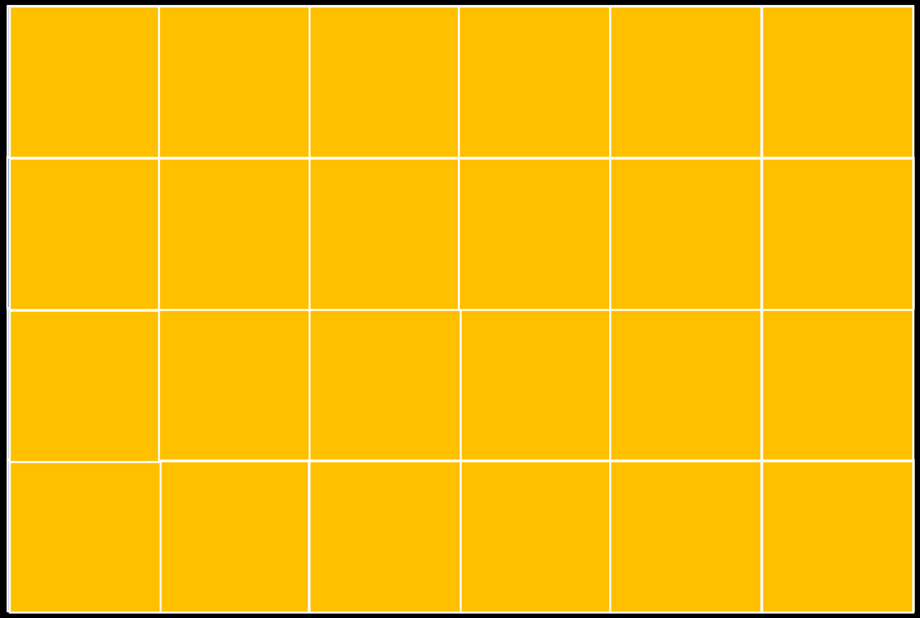
Field interpolation



How can each field variable vary over the element?

- Constant
- Element based
- Node based
- Point based

Generic temporal-spatial problem



- Loop over time
- Update current field parameters from previous (or initial) time step

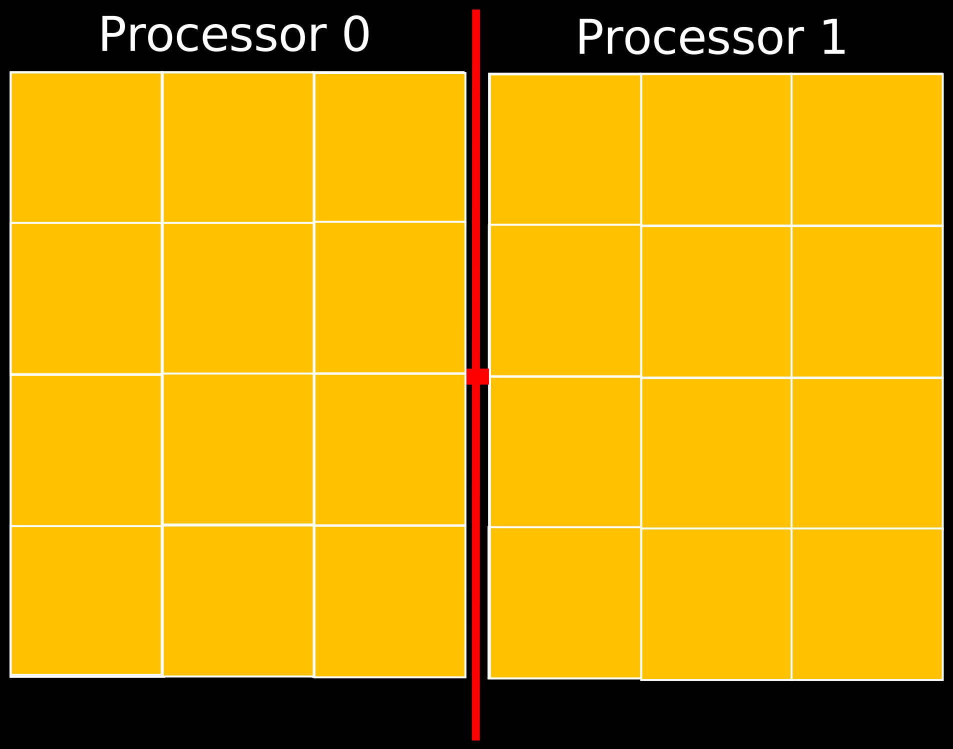
Loop over elements
- Integrate Cell Models
- Calculate element equations

Endloop

- Assemble global equation system
- Solve spatial system
- Calculate field parameters at next time step

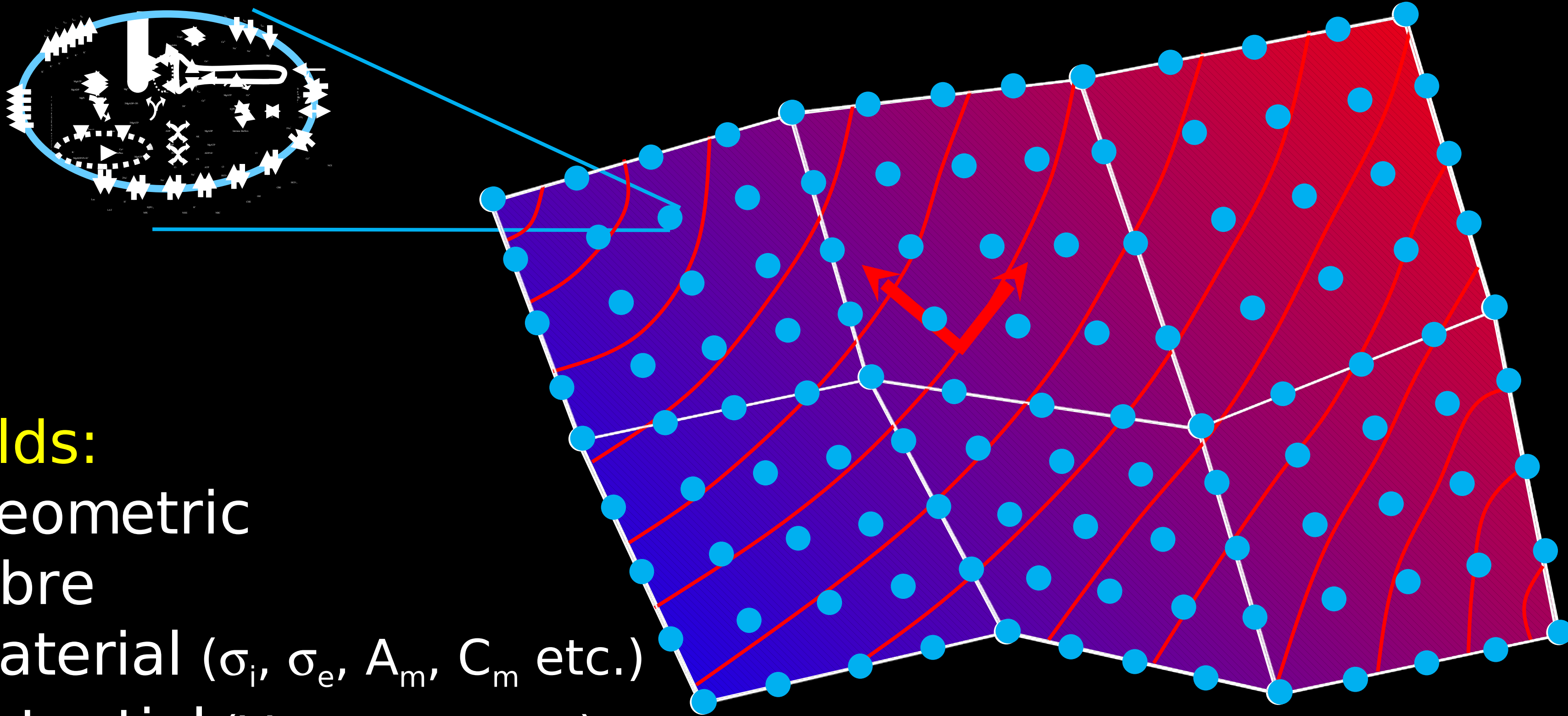
Endloop

Generic temporal-spatial problem (distributed)




```
Loop over time
    Update current field parameters
      from previous (or initial) time step
   Start ghost transfer of previous
      time step solution
   Loop over internal elements
      Integrate Cell Models
      Calculate element equations
   Endloop
   Wait until transfer has finished
   Loop over boundary+ghost elements
      Integrate Cell Models
      Calculate element equations
   Endloop
   Assemble global equation system
   Solve spatial system
   Calculate field parameters at next
      time step
Endloop
```

Problem/Field description



Fields:
- Geometric
- Fibre
- Material (σ_i, σ_e, A_m, C_m etc.)
- Potential (V_m, Φ_e, Φ_i etc.)

- Source (cell variables)
- Source material (cell parameters)

Source material field

- Want to be able to vary any cell parameter spatially
- Two options
  - Interpolate every parameter before evaluating a cell model.
    - Most general
    - Computationally expensive as most parameters are constant
  - Only allow certain parameters to vary and interpolate them

Fields Involved

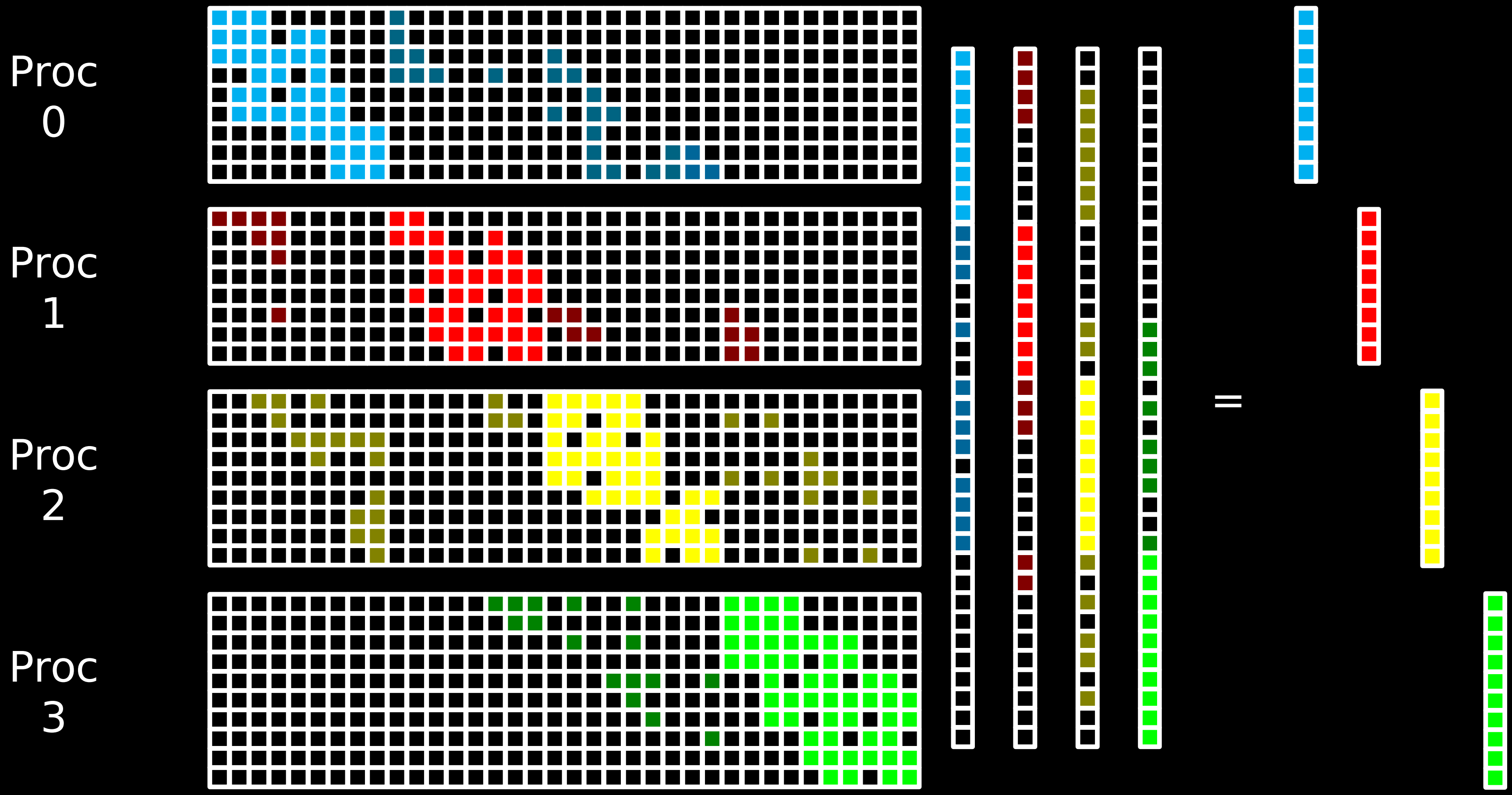
- Geometric field
- Fibre field
- Potential fields (V_m, Φ_e, Φ_i etc.)
- Material fields ($\boldsymbol{\sigma}_i$, $\boldsymbol{\sigma}_e$, A_m, C_m etc.)

- Source field (cell state variables)
- Constant source material field (cell parameters)
- Varying source material field (cell parameters)

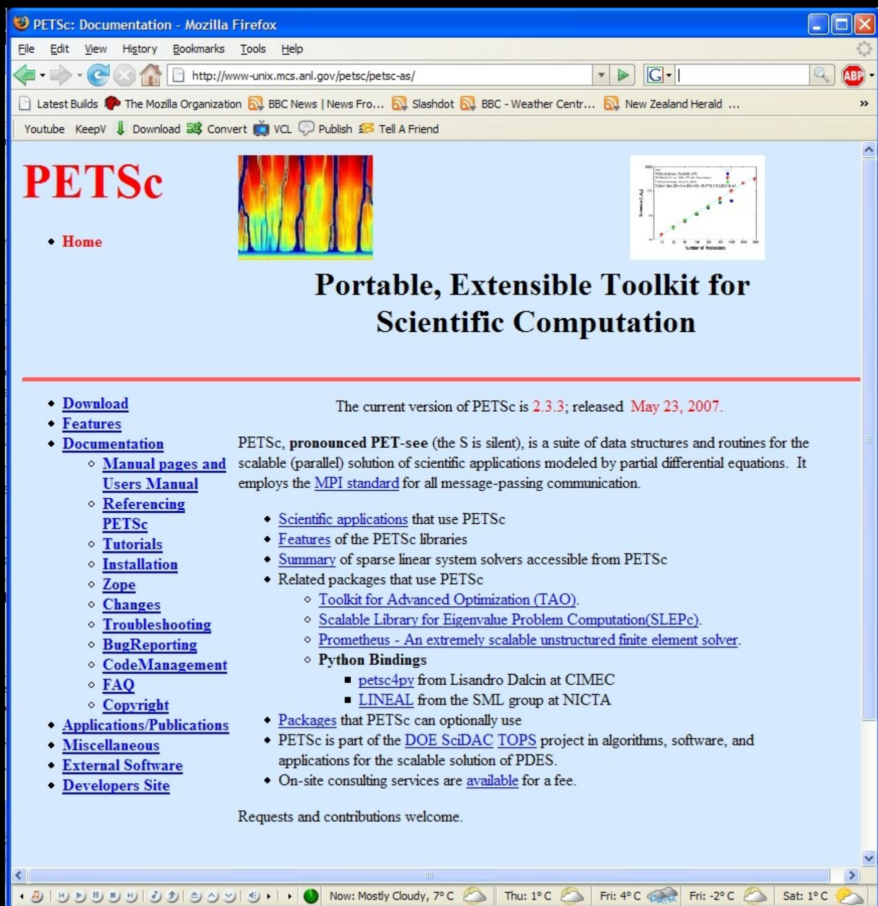
FieldML

CellML

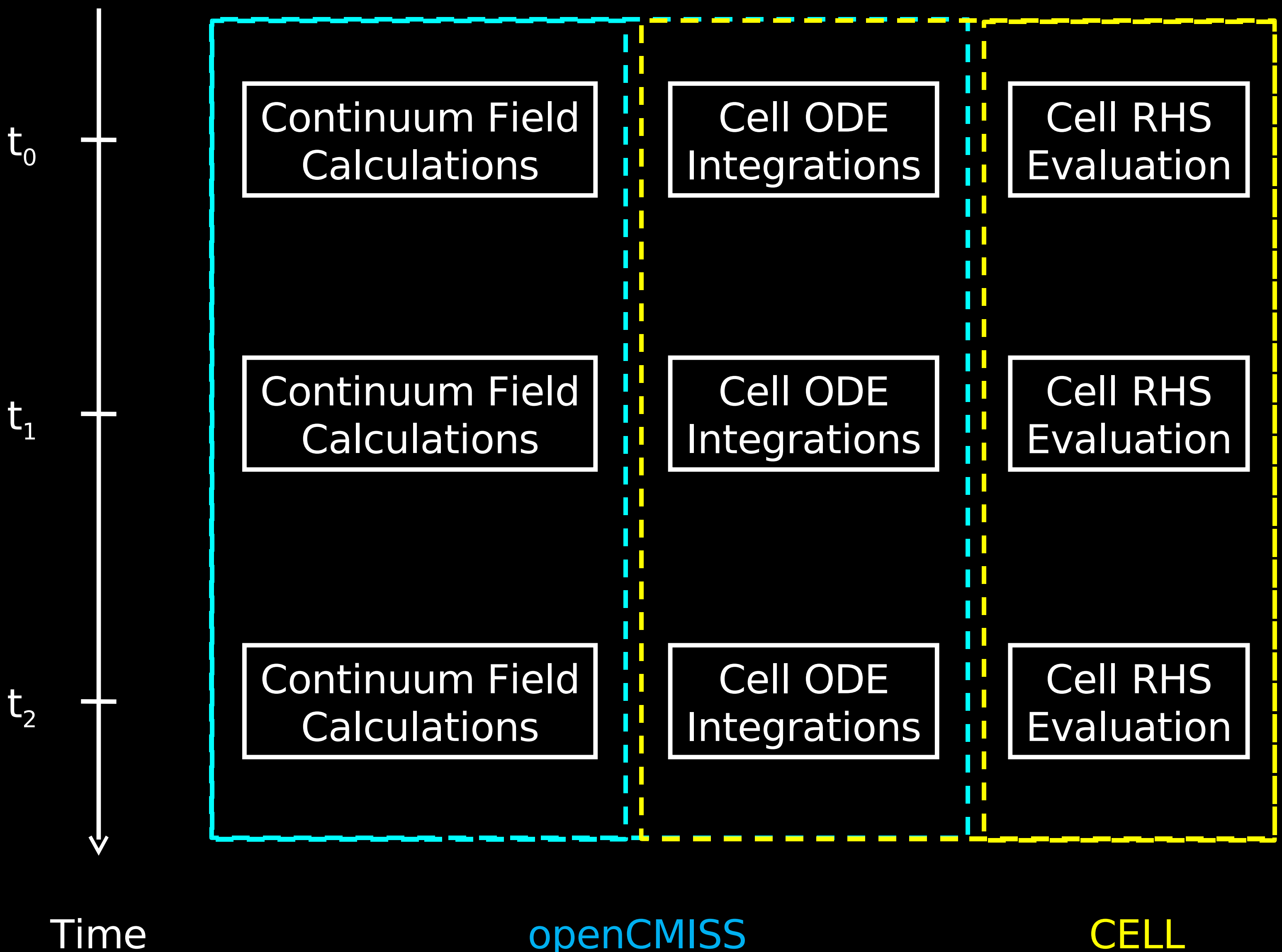
Matrix distribution



Distributed solvers



Cell Model Evaluation Scope



Acknowledgement

wellcome trust