

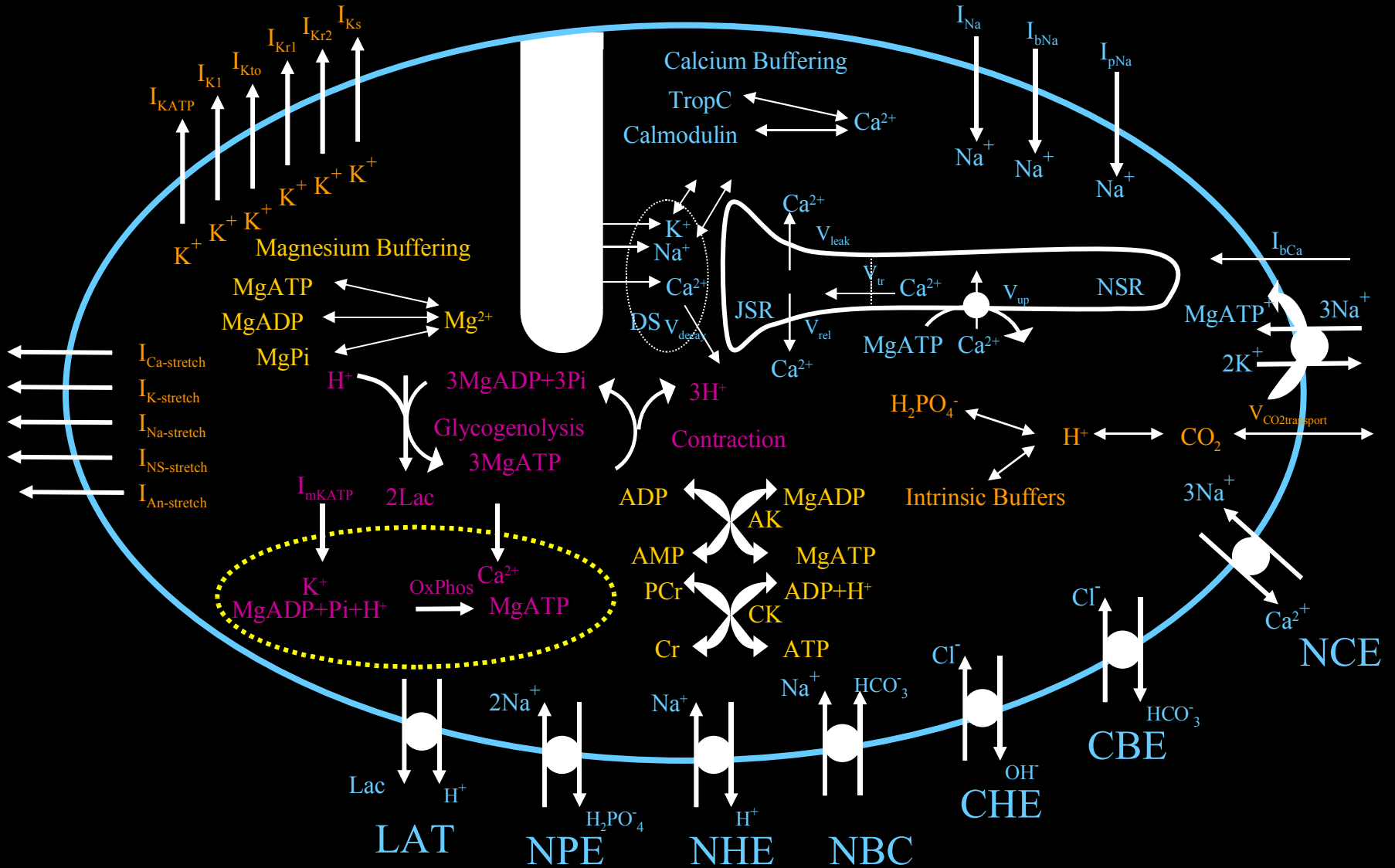
# CellML Workshop 2008 – openCMISS and FieldML/CellML

**Chris Bradley**

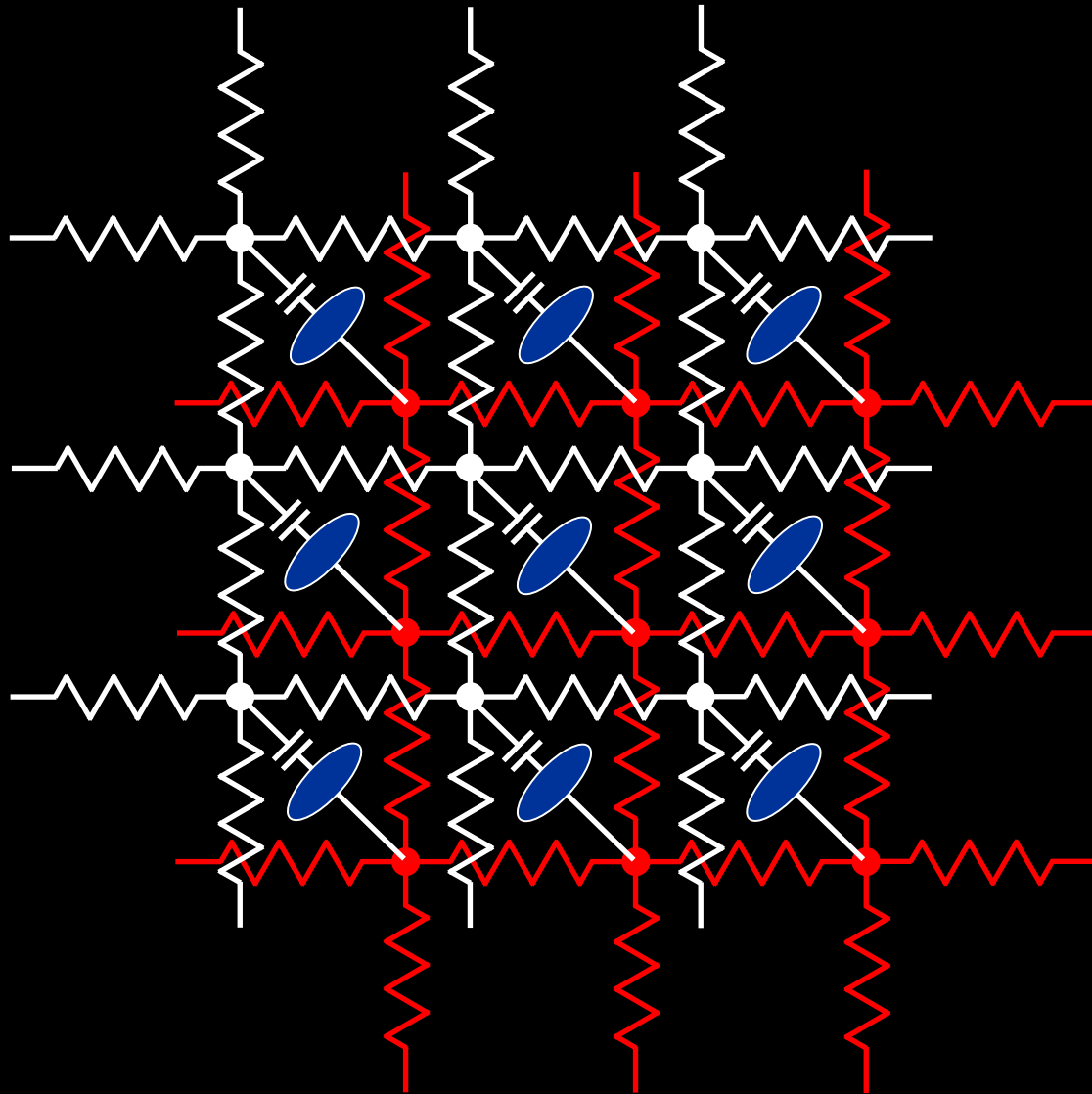



Department of Physiology,  
Anatomy and Genetics,  
University of Oxford

# Cellular processes



# Bidomain representation



Cell model = 

Extracellular Space

$\sigma_e, \Phi_e$

Cell Membrane

Intracellular Space

$\sigma_i, \Phi_i$

$I_{ion}$

# Bidomain equations

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

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

$$V_m = \Phi_e - \Phi_i$$

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
- $\Phi_e$  = extracellular potential
- $\Phi_i$  = intracellular potential
- $\sigma_e$  = intracellular conductivity tensor
- $\sigma_i$  = intracellular conductivity tensor

# BOTE problem size calculation

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

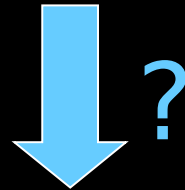
# Solution time calculation

- A 2.4 GHz Intel Core 2 can achieve  $\sim 1.7$  Gflops with linpack (best case!)
- To simulate 1 min real time on 1 processor would therefore take  $4.48 \times 10^7$  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

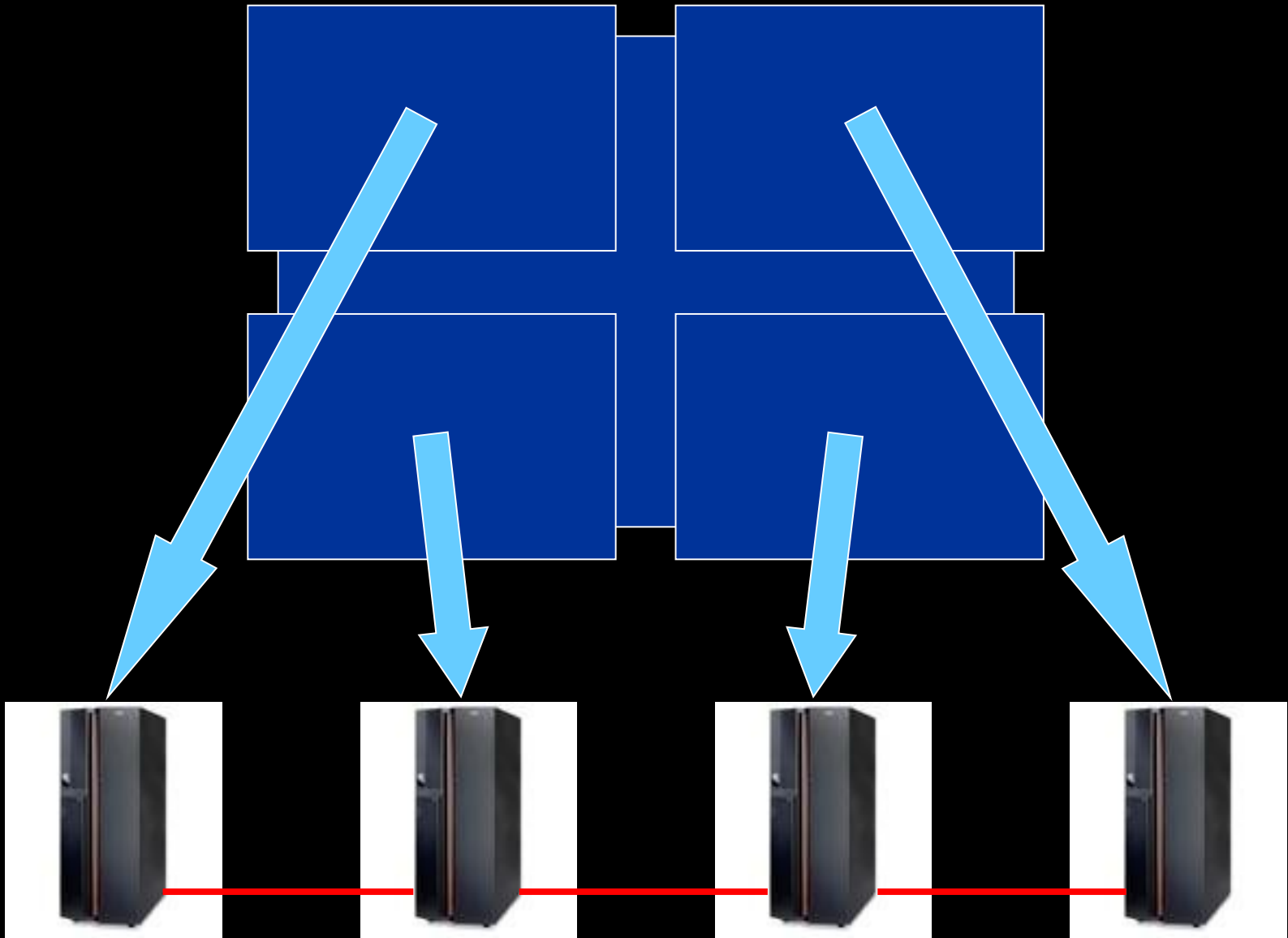
- Re-engineering of CMISS computational engine.
- Want
  - a library based approach to enable use in multiple applications.
  - Modular, easily extendable and programmable.
  - MPI based distributed memory + OpenMP + serial code
  - Easy to understand and program by novices.
- Open source on sourceforge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

# Parallelisation?

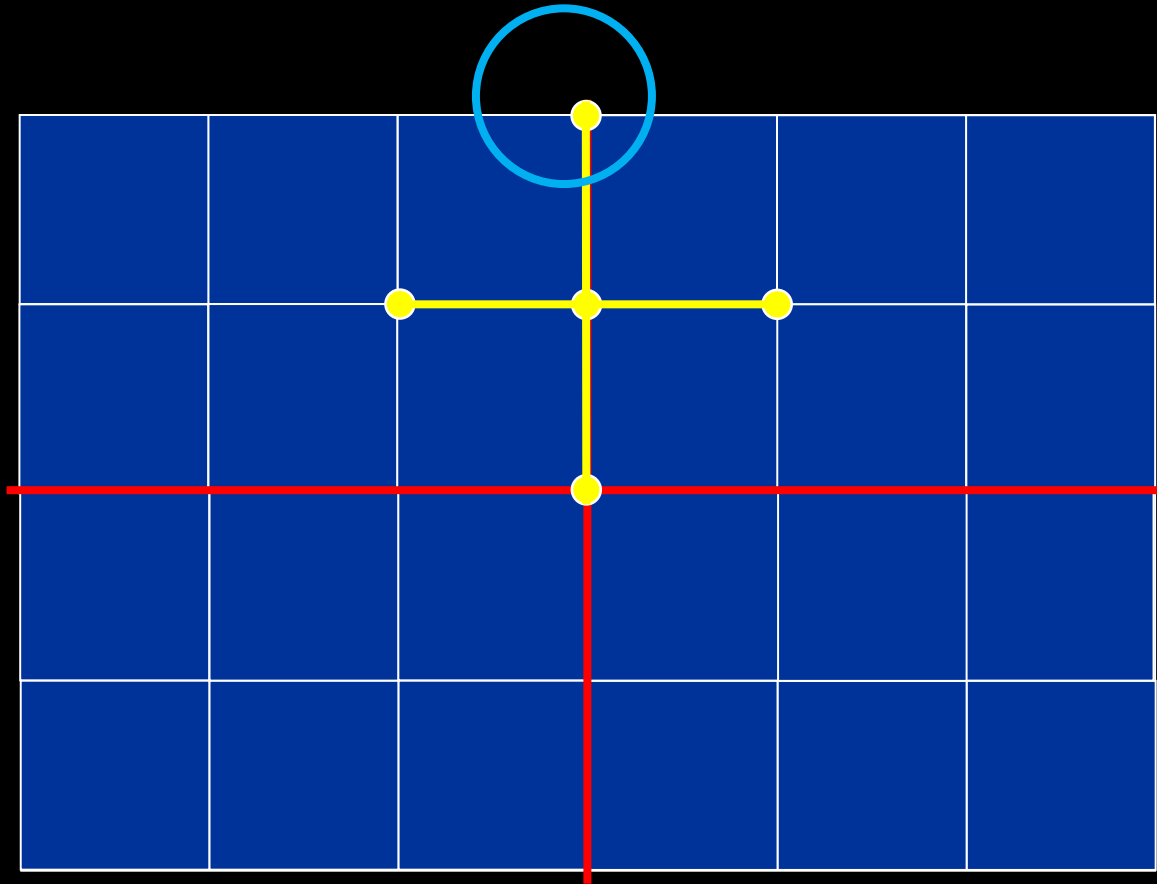




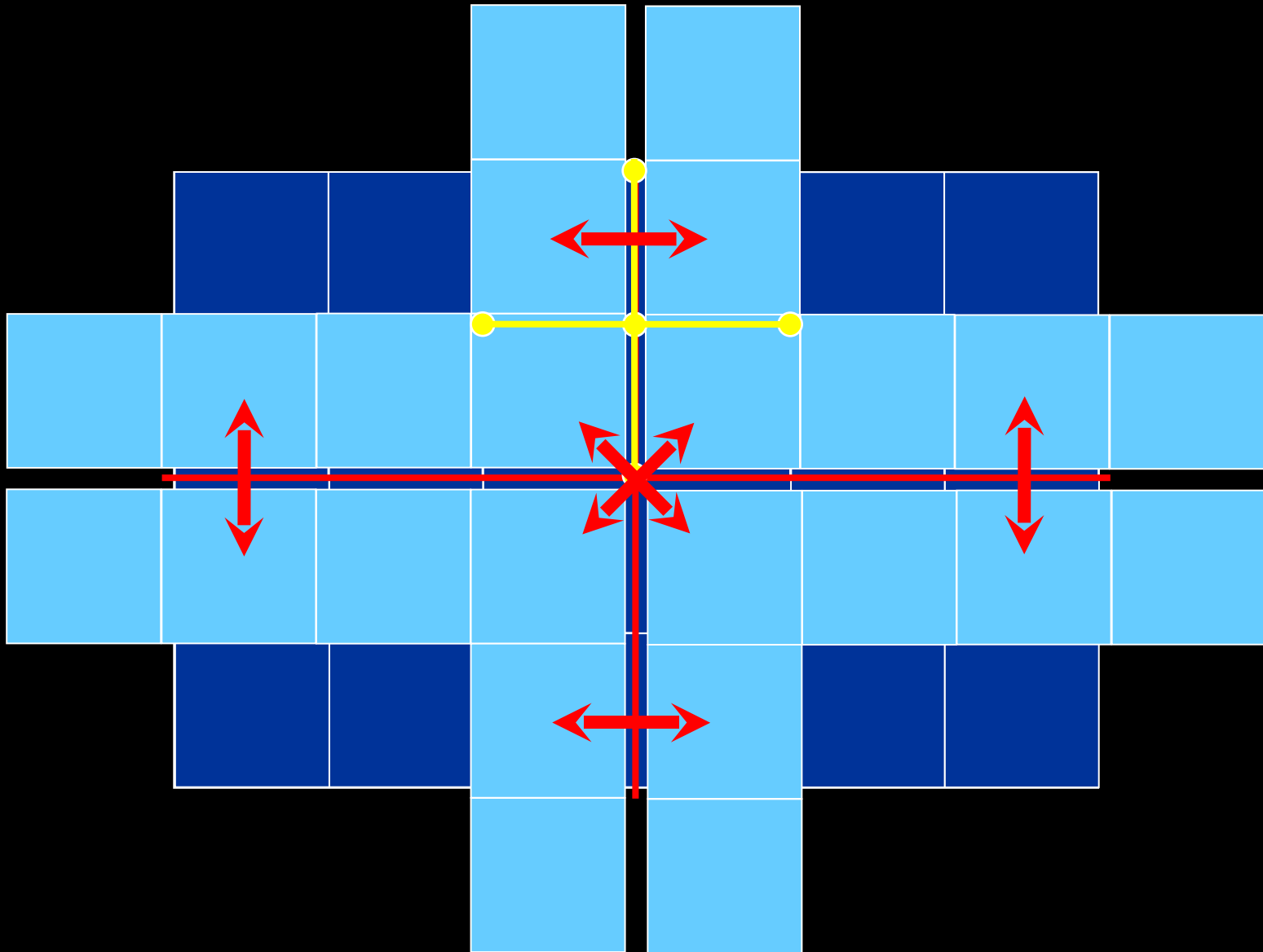
# Domain decomposition



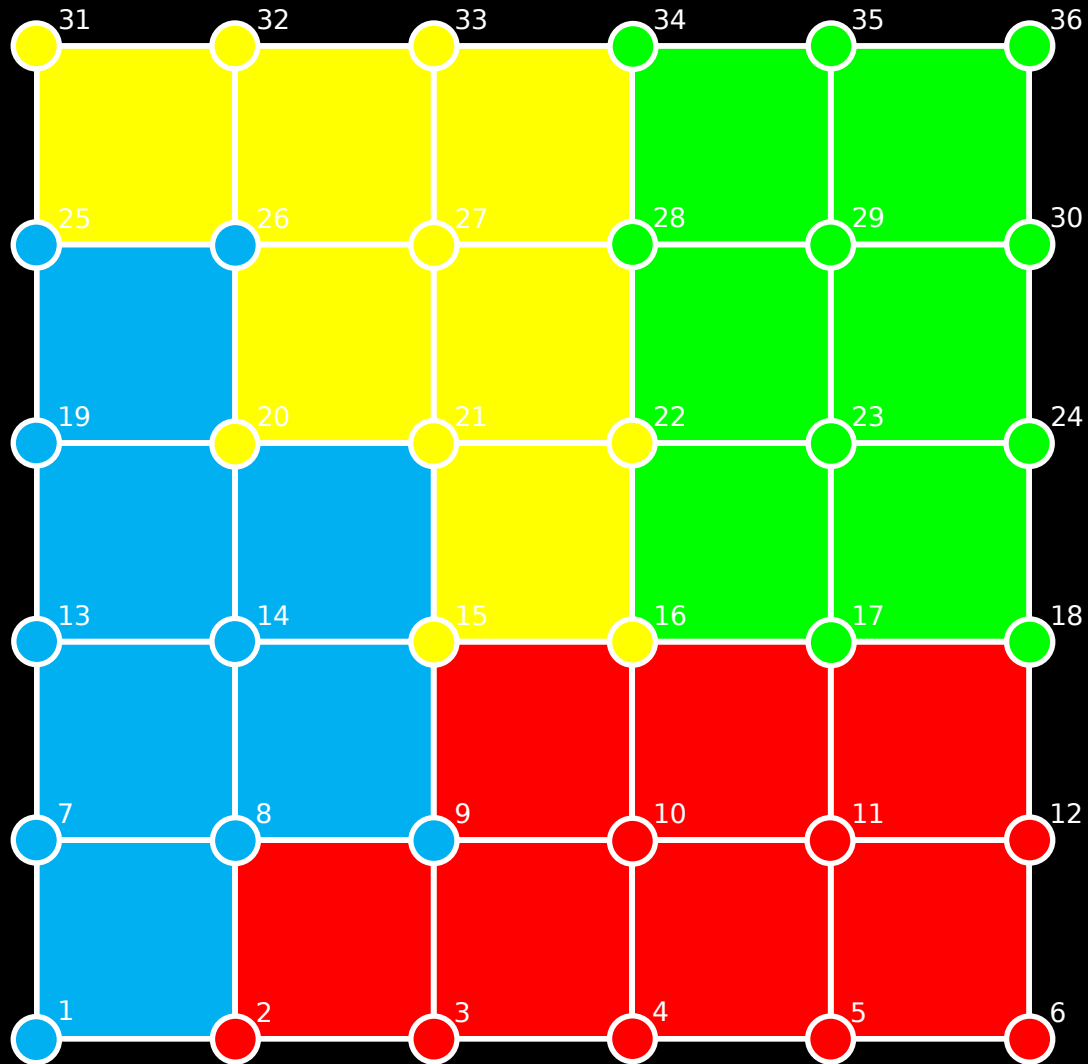
# Domain decomposition



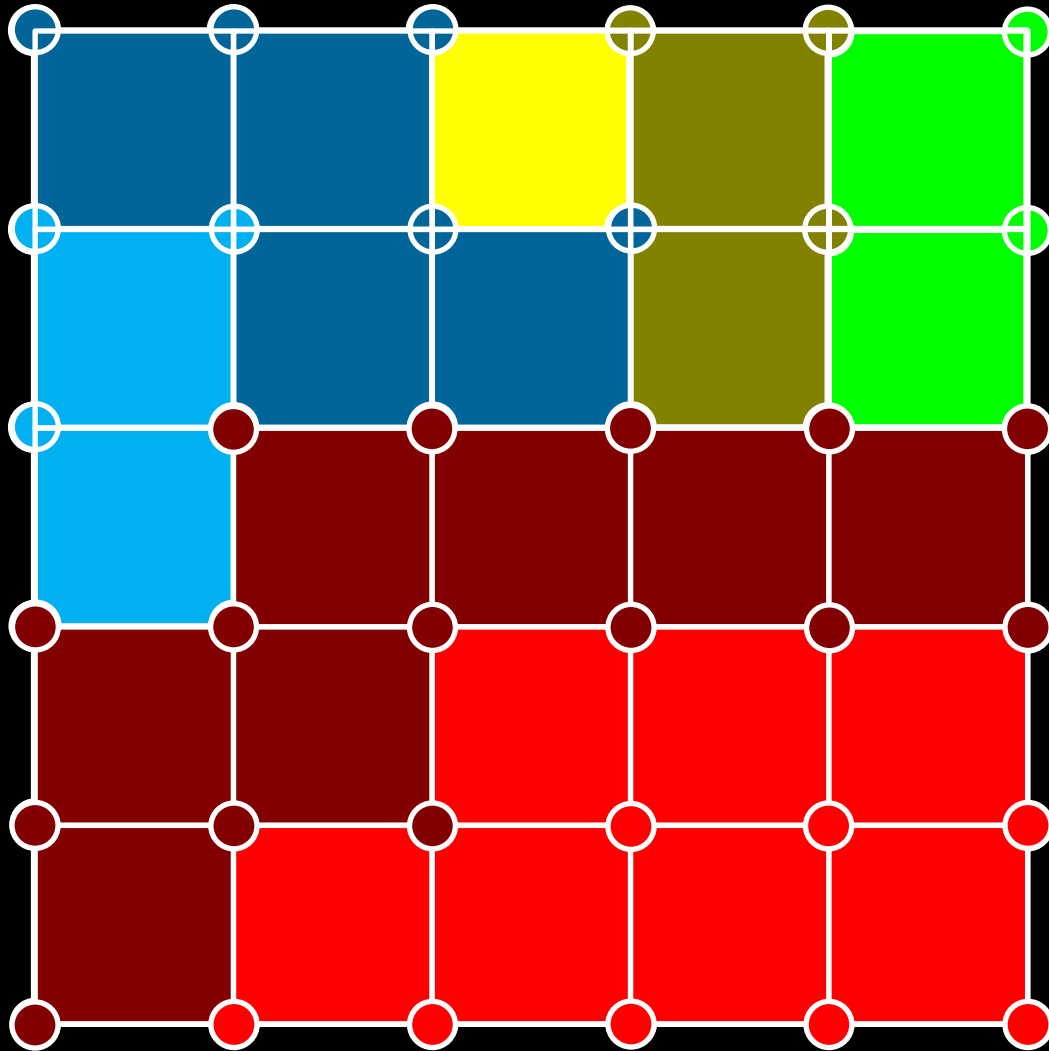
# Ghosting



# Mesh Decomposition



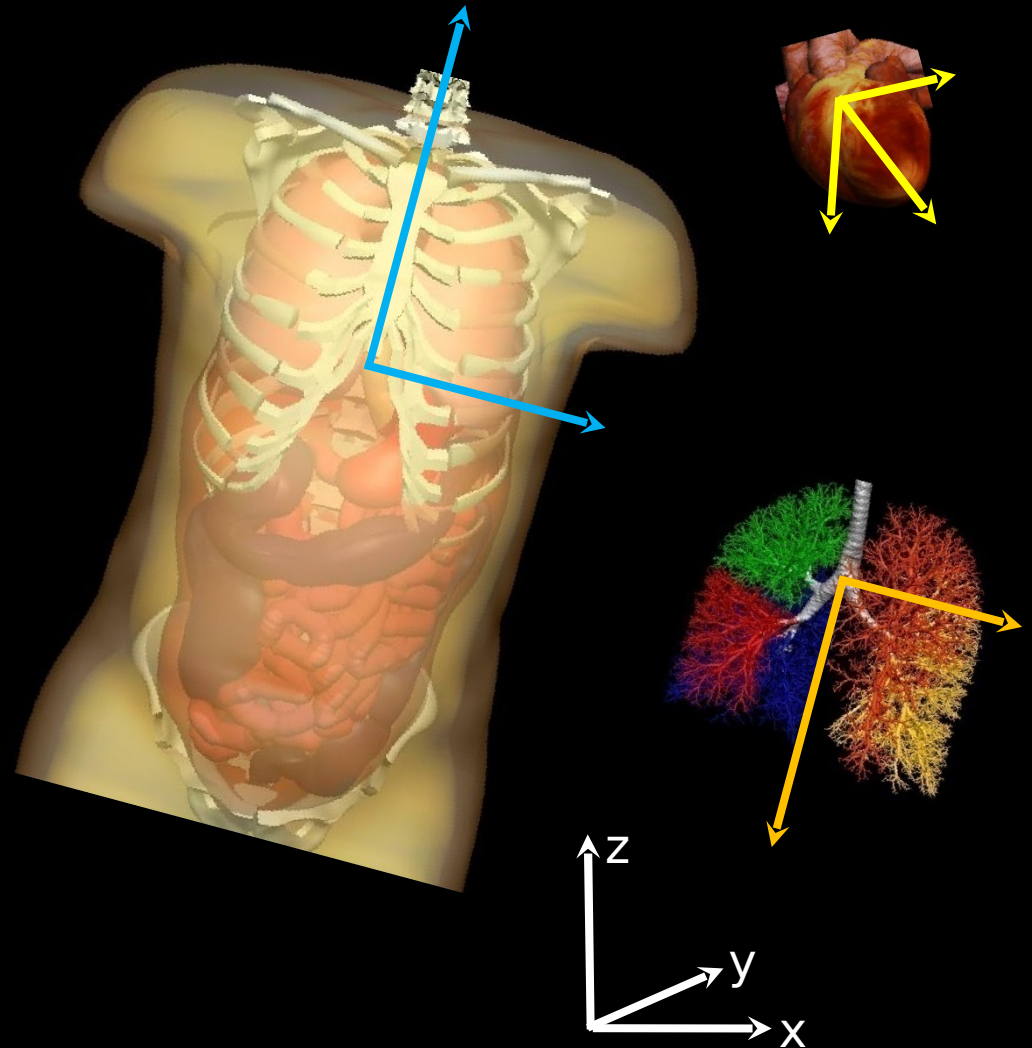
# Mesh Decomposition



# Regions

Regions contain:

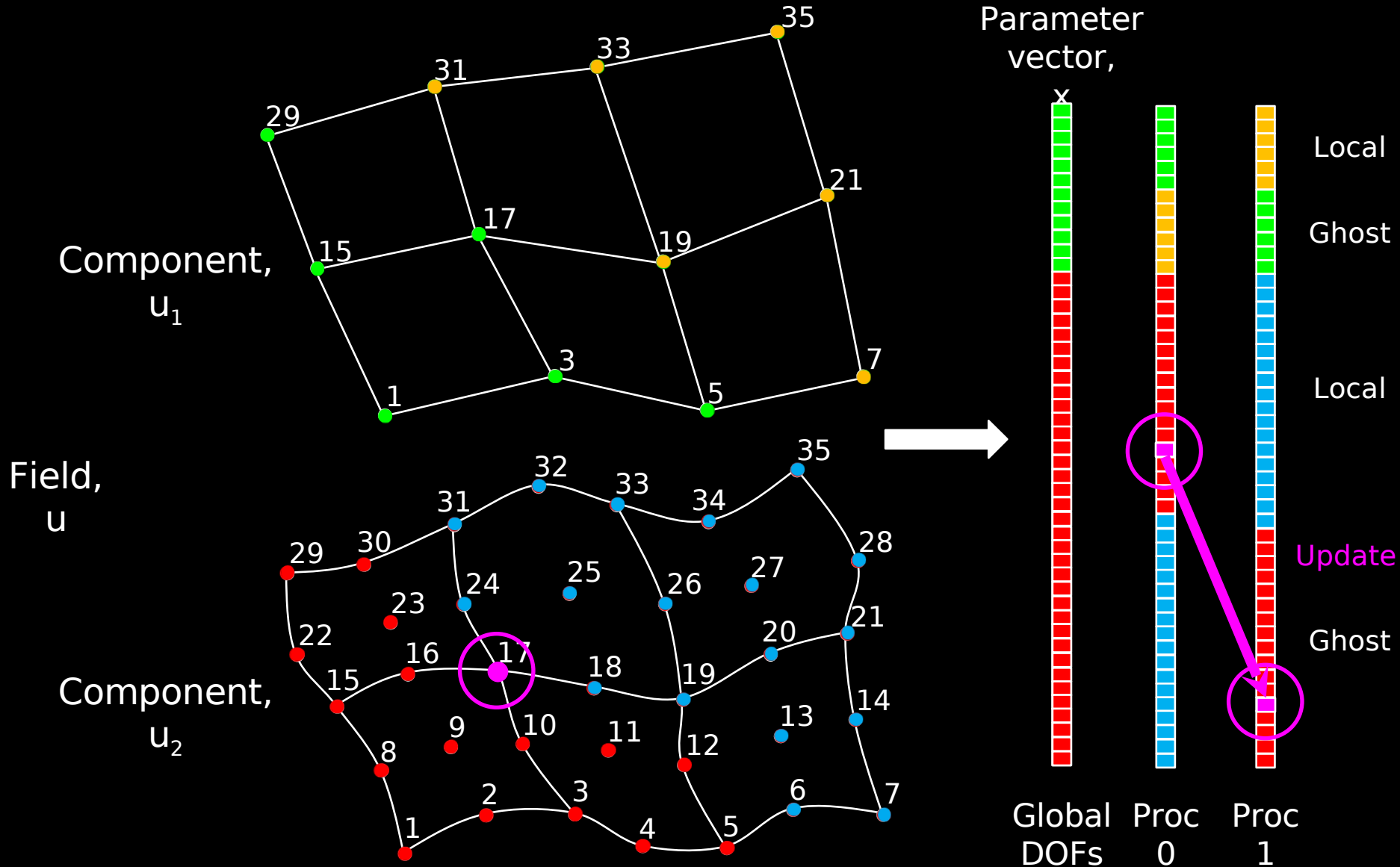
- Coordinate system
- Nodes
- Meshes
- Fields
- Problems
- Daughter and parent regions



# Fields

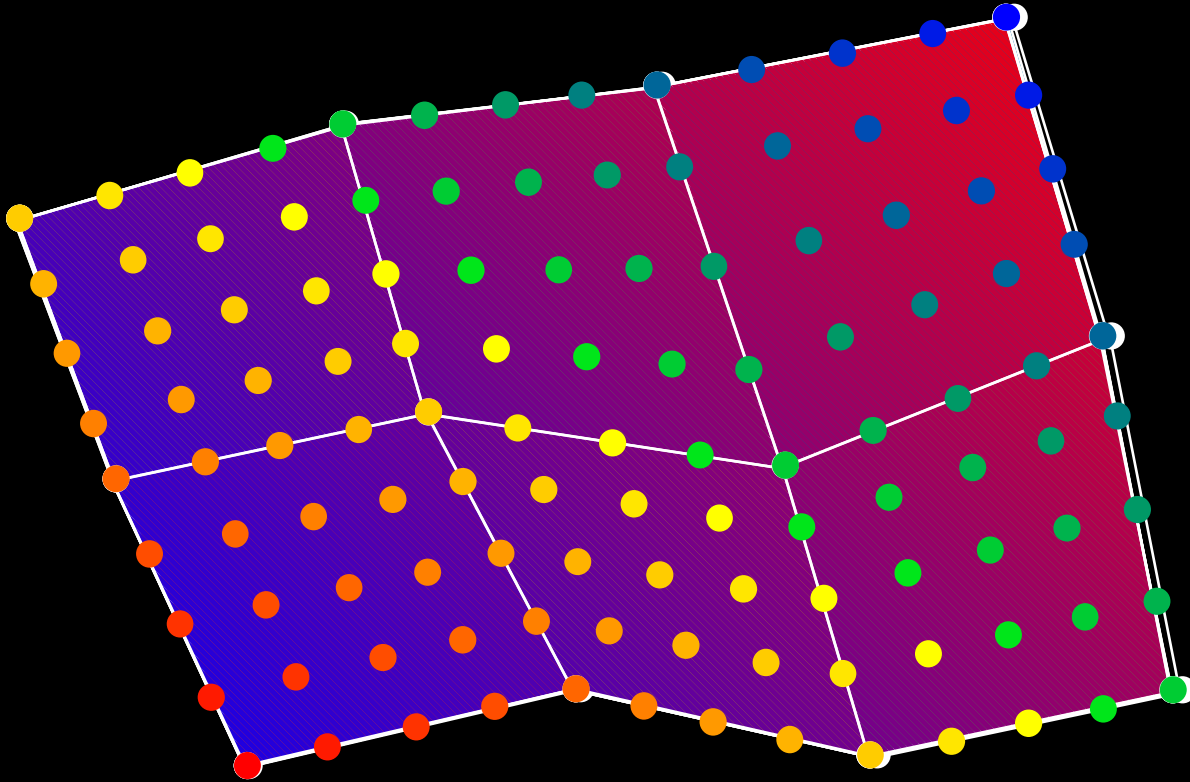
- Fields are the central object for storing information and framing the problem.
- Fields have a number of field variables i.e.,  $u$ ,  $\partial u/\partial n$ ,  $\partial u/\partial t$ ,  $\partial^2 u/\partial t^2$ .
- Each field variable has a number of components.
- A field is defined on a decomposed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

# Field parameter vector



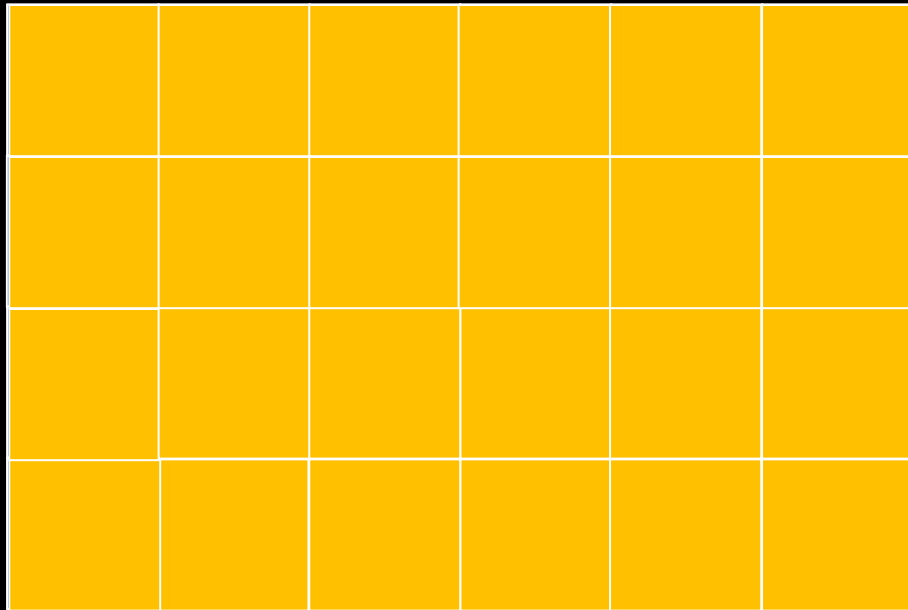


# Field interpolation



- How can each field be represented?
- Element based
  - Node based
  - Point based

# Generic temporal-spatial problem

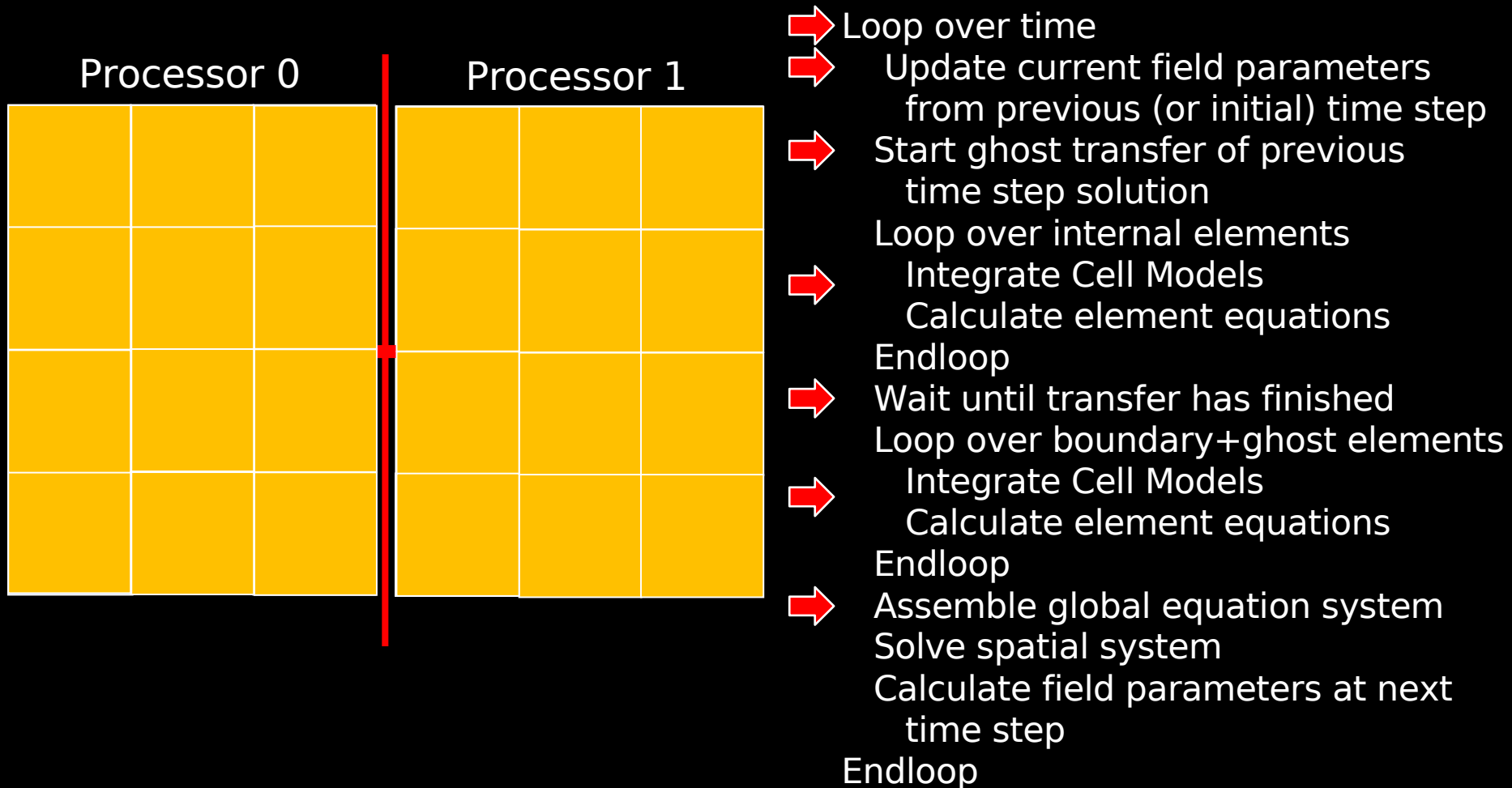


→ Loop over time  
→ Update current field parameters from previous (or initial) time step

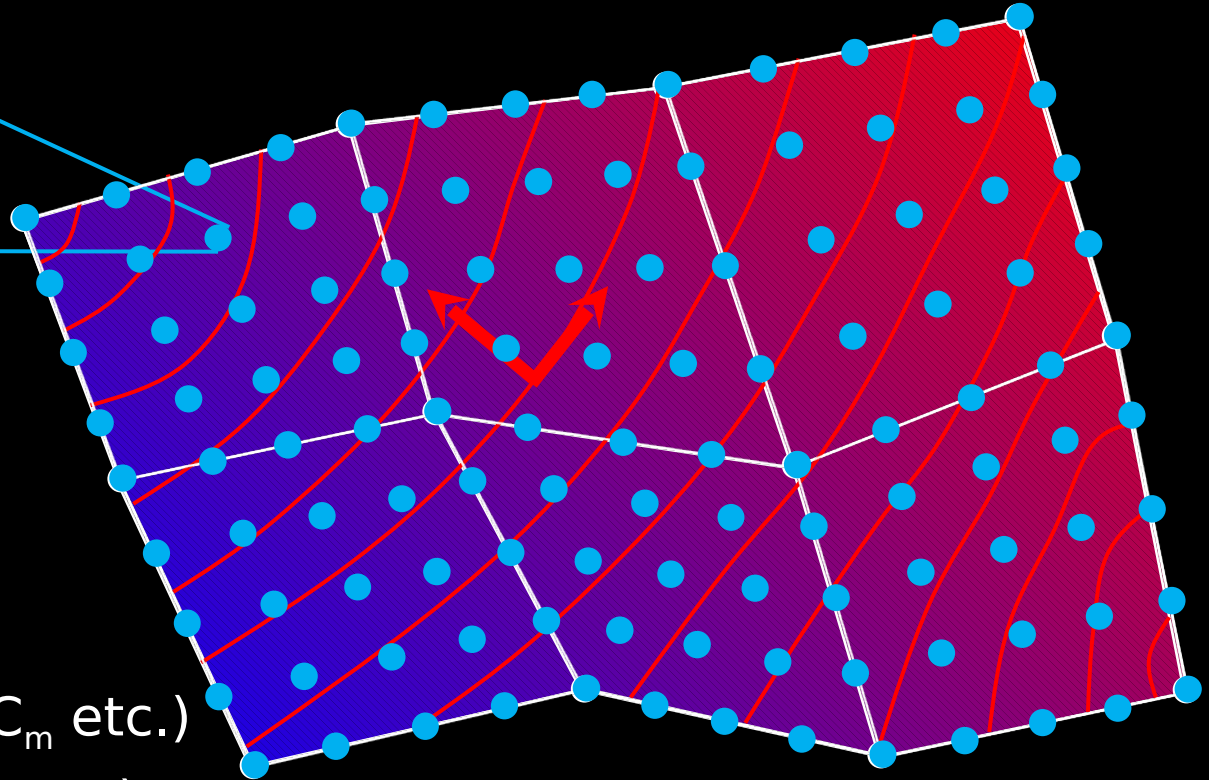
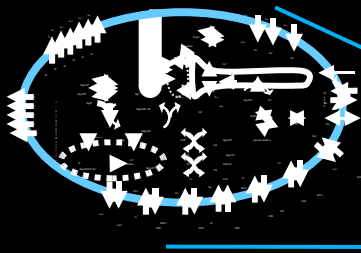
→ Loop over elements  
Integrate Cell Models  
Calculate element equations  
Endloop

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

# Generic temporal-spatial problem (distributed)



# Problem/Field description



## Fields:

- Geometric
- Fibre
- Material ( $\sigma_i$ ,  $\sigma_e$ ,  $A_m$ ,  $C_m$  etc.)
- Potential ( $V_m$ ,  $\Phi_e$ ,  $\Phi_i$  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

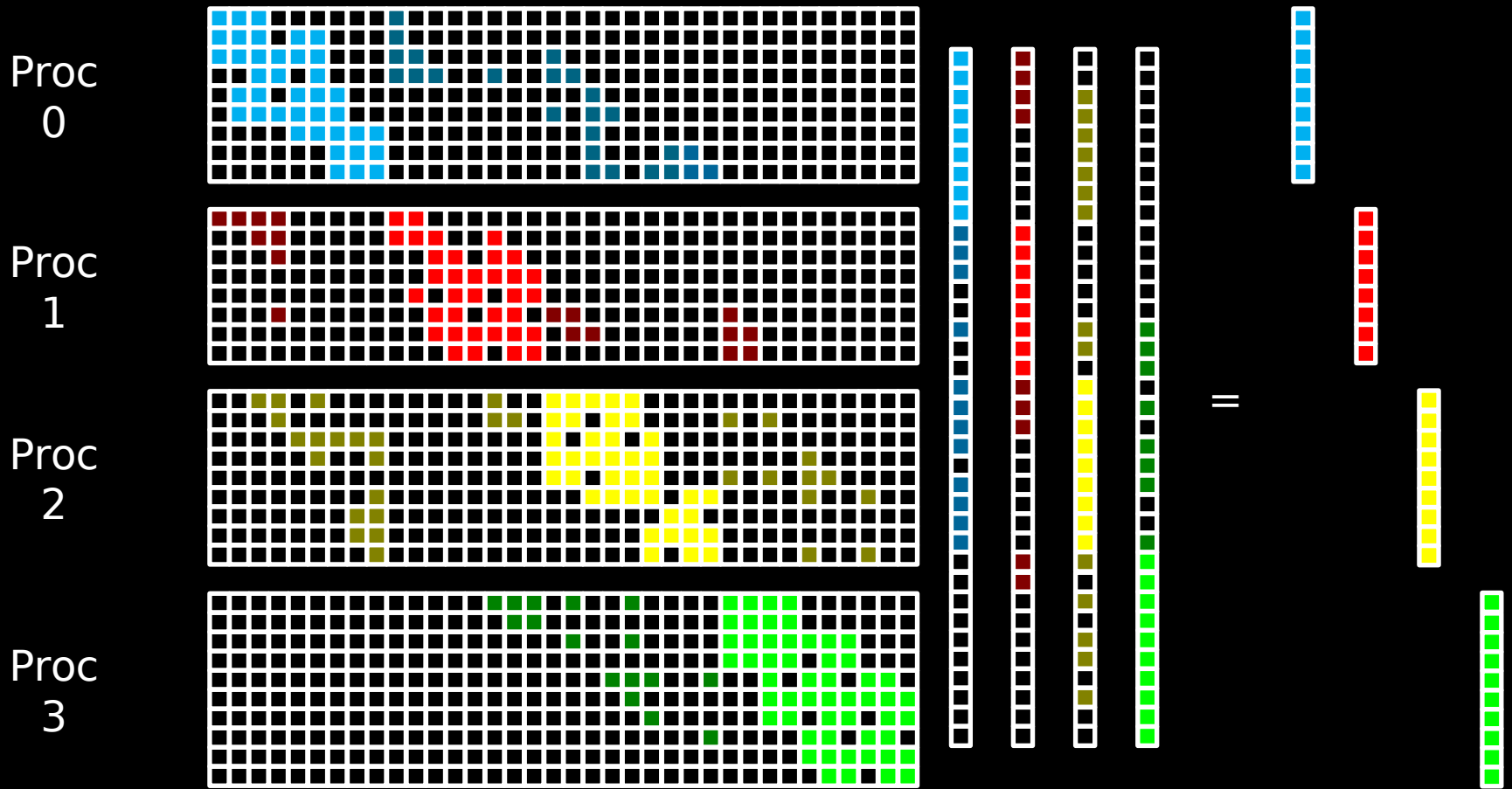
# Fields Involved

- Geometric field
- Fibre field
- Potential fields ( $V_m$ ,  $\Phi_e$ ,  $\Phi_i$  etc.)
- Material fields ( $\sigma_i$ ,  $\sigma_e$ ,  $A_m$ ,  $C_m$  etc.)
  
- Source field (cell state variables)
- Constant source material field (cell parameters)
- Varying source material field (cell parameters)

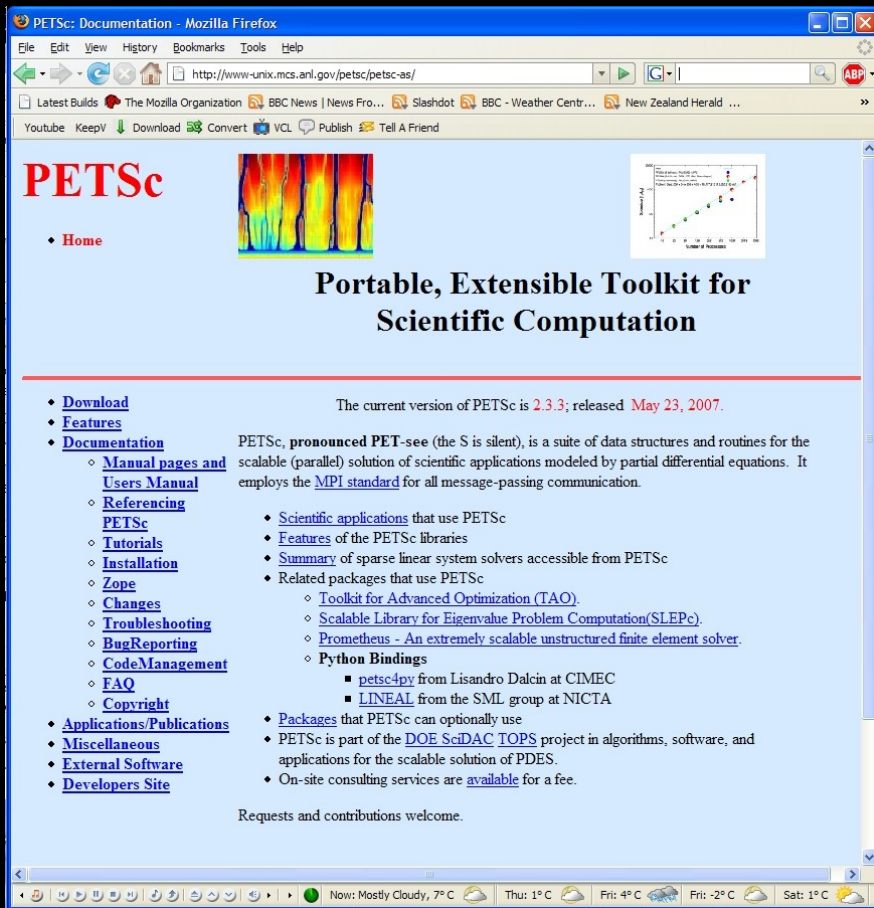
FieldML

CellML

# Matrix distribution

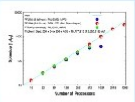
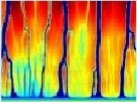


# Distributed solvers



**PETSc**

• Home



## Portable, Extensible Toolkit for Scientific Computation

The current version of PETSc is 2.3.3; released **May 23, 2007**.

PETSc, pronounced PET-see (the S is silent), is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. It employs the [MPI standard](#) for all message-passing communication.

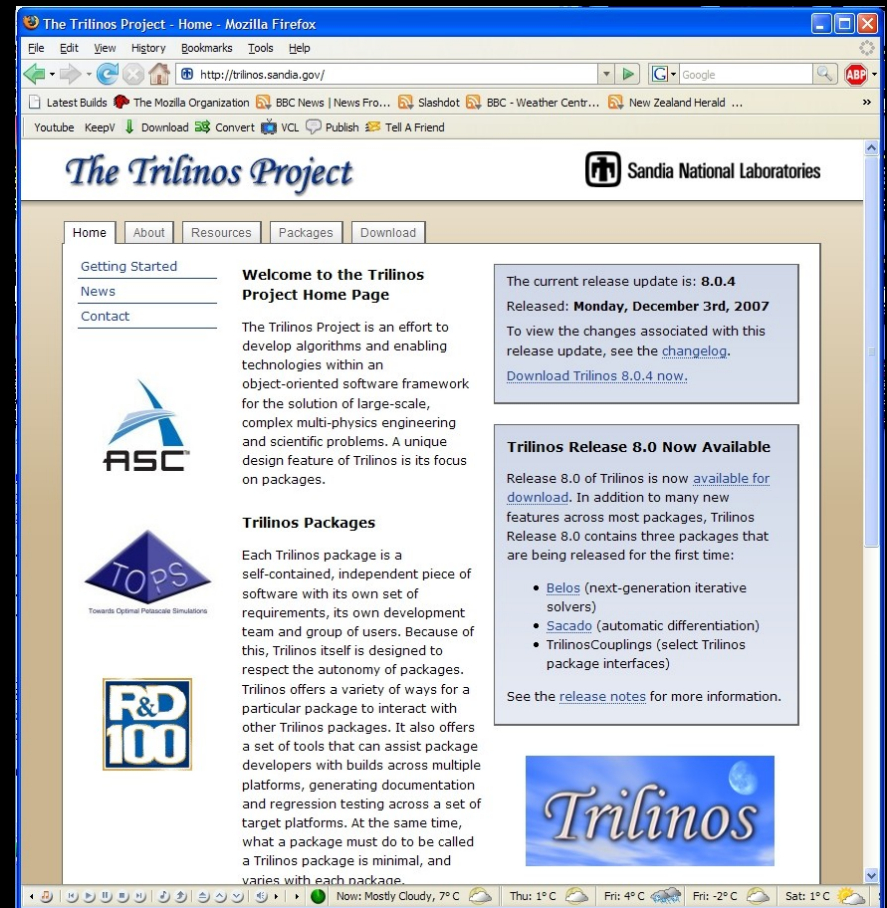
- [Scientific applications](#) that use PETSc
- [Features](#) of the PETSc libraries
- [Summary](#) of sparse linear system solvers accessible from PETSc
- Related packages that use PETSc
  - [Toolkit for Advanced Optimization \(TAO\)](#).
  - [Scalable Library for Eigenvalue Problem Computation \(SLEPC\)](#).
  - [Prometheus - An extremely scalable unstructured finite element solver](#).
  - [Python Bindings](#)
    - [petsc4py](#) from Lisandro Dalcin at CIMEC
    - [LINEAL](#) from the SML group at NICTA
- [Packages](#) that PETSc can optionally use
- PETSc is part of the [DOE SciDAC TOPS](#) project in algorithms, software, and applications for the scalable solution of PDES.
- On-site consulting services are [available](#) for a fee.

Requests and contributions welcome.

Navigation: [Download](#), [Features](#), [Documentation](#)

- [Manual pages and Users Manual](#)
- [Referencing](#)
- [PETSc](#)
- [Tutorials](#)
- [Installation](#)
- [Zope](#)
- [Changes](#)
- [Troubleshooting](#)
- [BugReporting](#)
- [CodeManagement](#)
- [FAQ](#)
- [Copyright](#)

- [Applications/Publications](#)
- [Miscellaneous](#)
- [External Software](#)
- [Developers Site](#)



## The Trilinos Project

Sandia National Laboratories

Home | About | Resources | Packages | Download

Getting Started  
News  
Contact

### Welcome to the Trilinos Project Home Page

The Trilinos Project is an effort to develop algorithms and enabling technologies within an object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific problems. A unique design feature of Trilinos is its focus on packages.

### Trilinos Packages

Each Trilinos package is a self-contained, independent piece of software with its own set of requirements, its own development team and group of users. Because of this, Trilinos itself is designed to respect the autonomy of packages. Trilinos offers a variety of ways for a particular package to interact with other Trilinos packages. It also offers a set of tools that can assist package developers with builds across multiple platforms, generating documentation and regression testing across a set of target platforms. At the same time, what a package must do to be called a Trilinos package is minimal, and varies with each package.


The current release update is: **8.0.4**  
Released: **Monday, December 3rd, 2007**  
To view the changes associated with this release update, see the [changelog](#).  
[Download Trilinos 8.0.4 now.](#)

### Trilinos Release 8.0 Now Available

Release 8.0 of Trilinos is now [available for download](#). In addition to many new features across most packages, Trilinos Release 8.0 contains three packages that are being released for the first time:

- [Belos](#) (next-generation iterative solvers)
- [Sacado](#) (automatic differentiation)
- [TrilinosCouplings](#) (select Trilinos package interfaces)

See the [release notes](#) for more information.

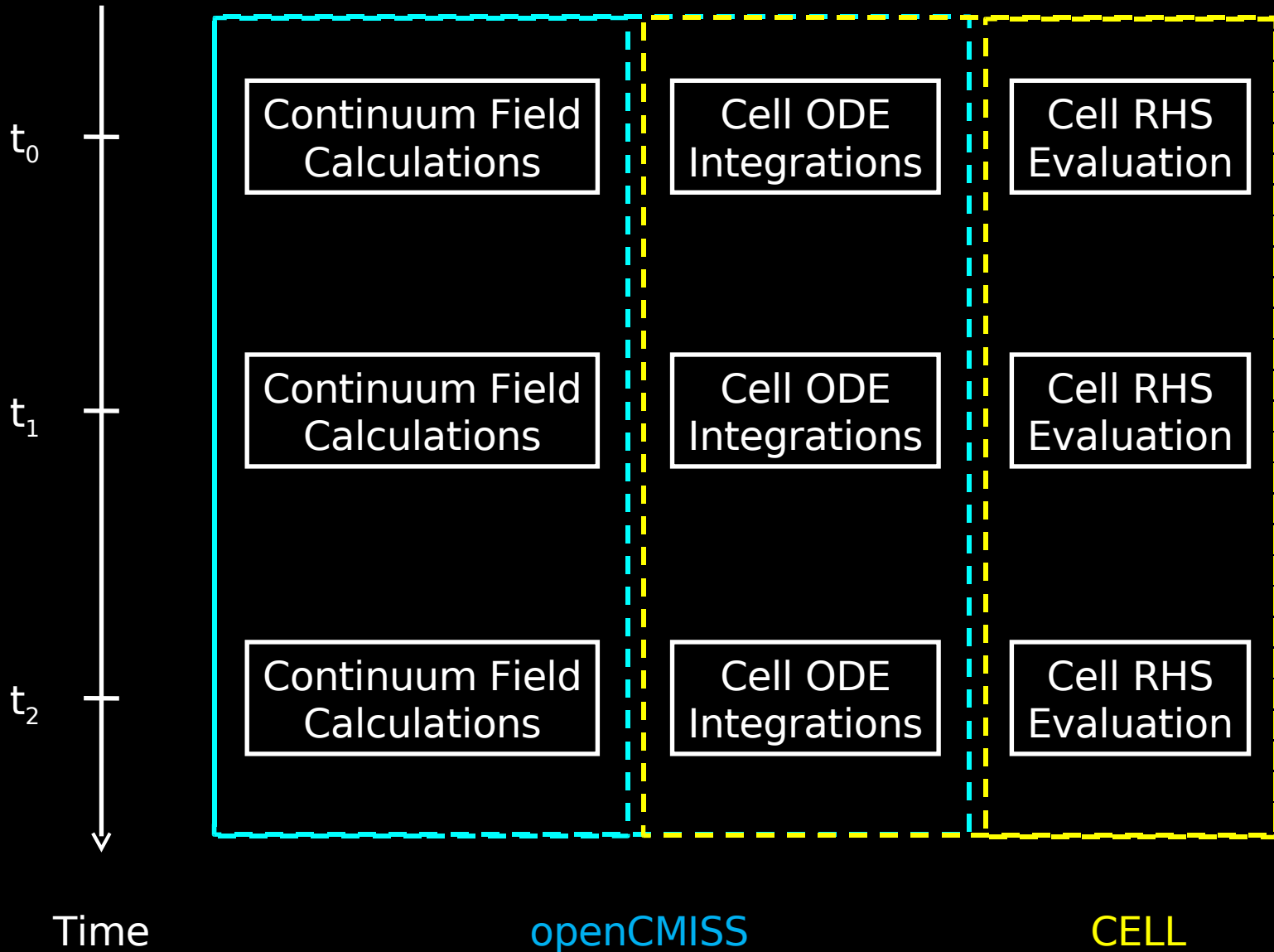


Navigation: [ASC](#), [TOPS](#), [R&D 100](#)

Now: Mostly Cloudy, 7° C | Thu: 1° C | Fri: 4° C | Fri: -2° C | Sat: 1° C



# Cell Model Evaluation Scope



# Acknowledgement

**wellcome**trust