

# Numerical Modelling in Python

Lutz Gross

Earth Systems Science Computational Center  
University of Queensland

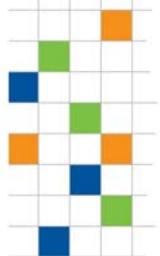
[l.gross@uq.edu.au](mailto:l.gross@uq.edu.au)

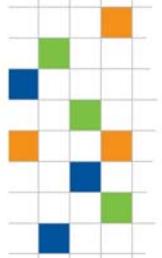
# People Involved

- K. Steuben (UQ)
- B. Cumming (UQ)
- E. Thorne (UQ)
- M. Davies (UQ)
- H. Muhlhaus (UQ)
- J. Smillie (UQ)
- R. Woodcock (CSIRO EM)
- P. Hornby (CSIRO EM)

# Outline

- Motivation: Geoscience Applications
- esys.escript
- Example
- Model Framework
- Summary

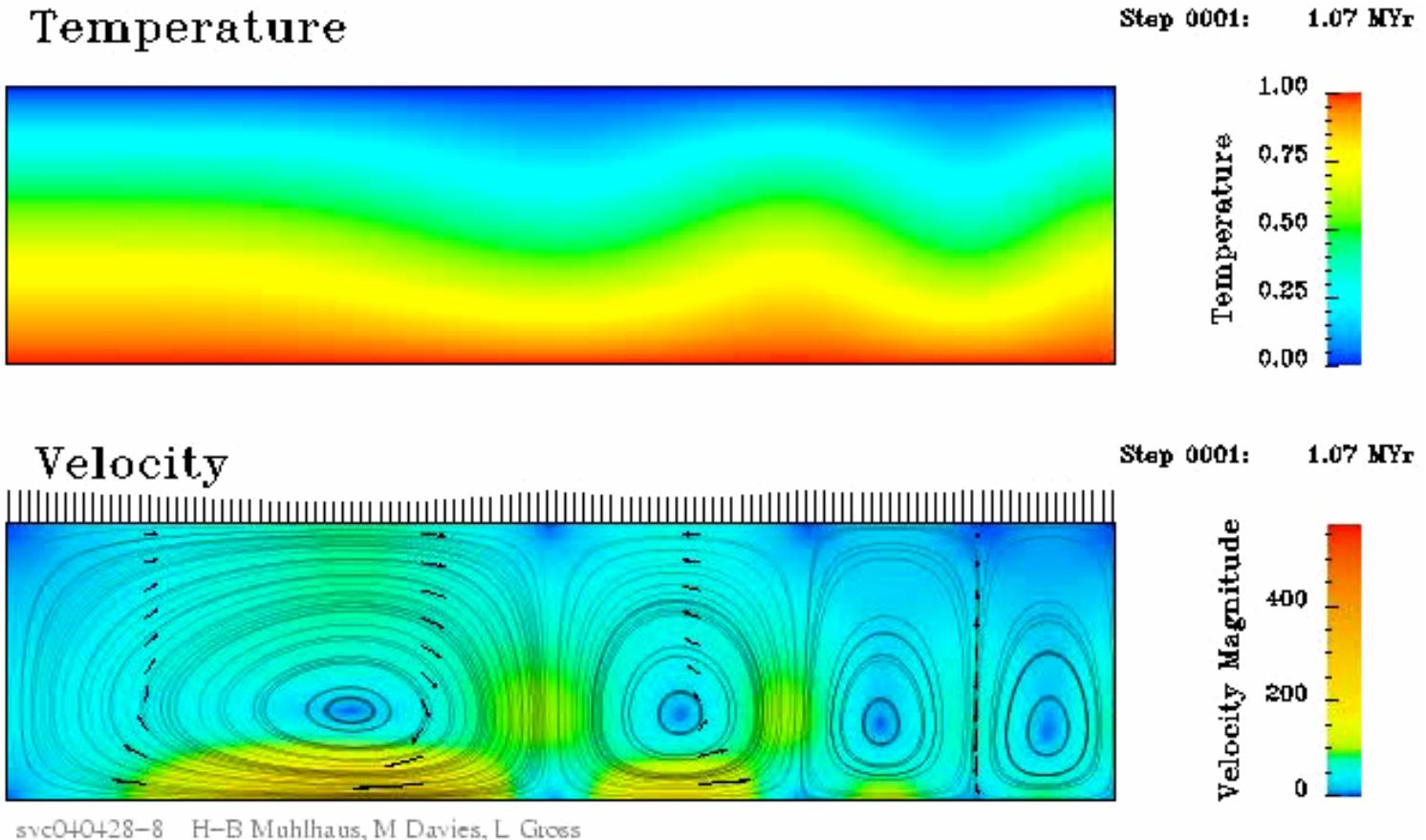




# Application: Earth mantle Convection

# Earth Mantle Convection

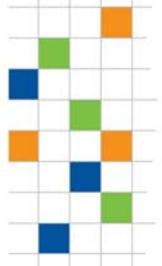
Muhlhaus, Davies, Bourgoine



June 06 / 5

# Basic Convection Model

Velocity  $v_i$ , pressure  $p$ , temperature  $T$


$$-\left(v_{j,i} + v_{i,j}\right)_{,j} - p_{,i} = -\omega T \delta_{i3}$$

$$-v_{i,i} = 0$$

$$T_{,t} + v_i T_{,i} - (T_{,i})_{,i} = 0$$

# Anisotropic Convection Model

$$-(\eta(v_{i,j} + v_{j,i}))_{,j} - p_{,i} + \omega T \delta_{3i} + ((\eta_s - \eta) \Lambda_{ijkl} v_{k,l})_{,j} = 0$$

$$-v_{k,k} = 0$$

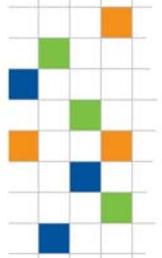
$$-(\kappa T_{,j})_{,j} + \rho c_p (\dot{T} + v_j T_{,j}) + \alpha \eta (v_{i,k} + v_{k,i})^2 - (\kappa_N n_i n_j T_{,k})_{,j} = 0$$

$$\Lambda_{ijkl} = \left( \frac{1}{2} (n_i n_k \delta_{lj} + n_j n_k \delta_{il} + n_i n_l \delta_{kj} + n_j n_l \delta_{ik}) - 2 n_i n_j n_k n_l \right)$$

director transition :  $\dot{n}_i = -v_{j,i} n_j$

$$\text{Viscosity : } \eta = A e^{a \left( \frac{1}{T} - \frac{1}{T_{ref}} \right)}$$

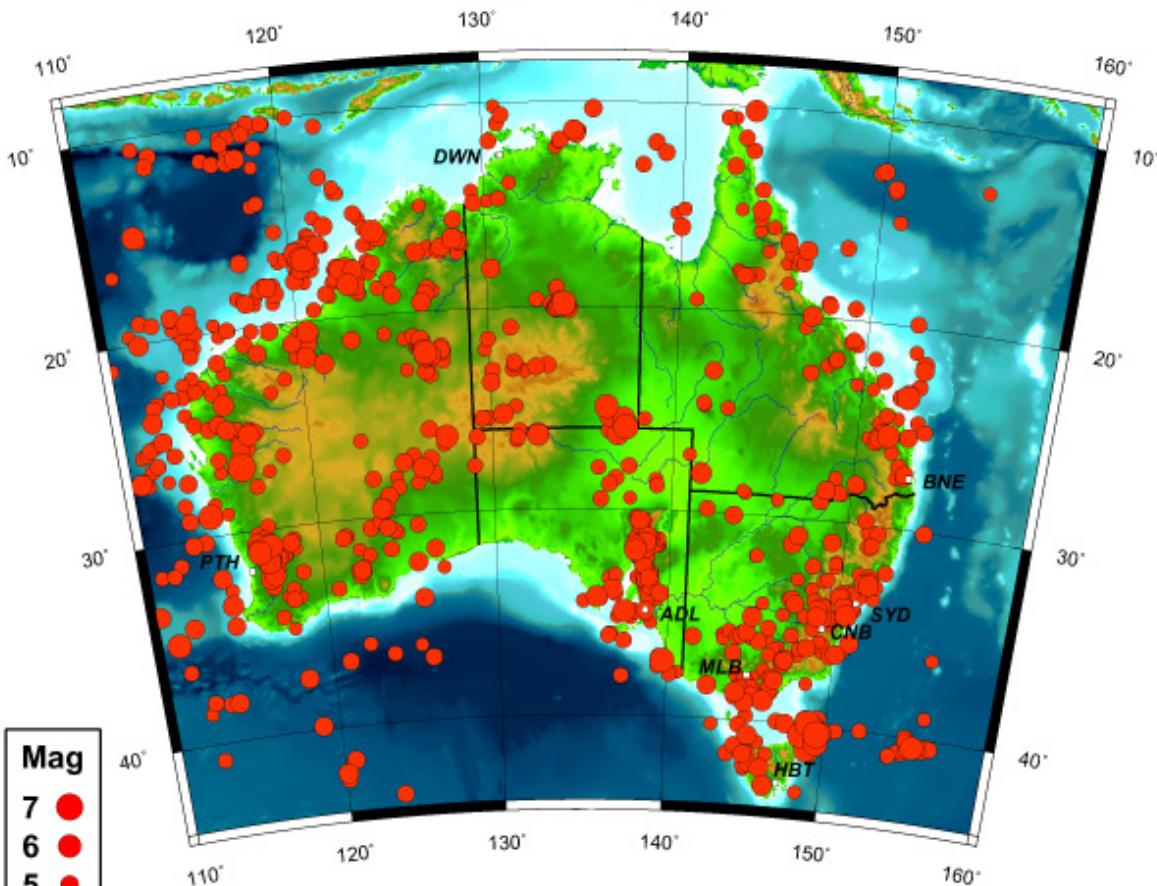
June 06 / 7



# Application: Earthquake Modelling

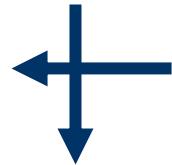
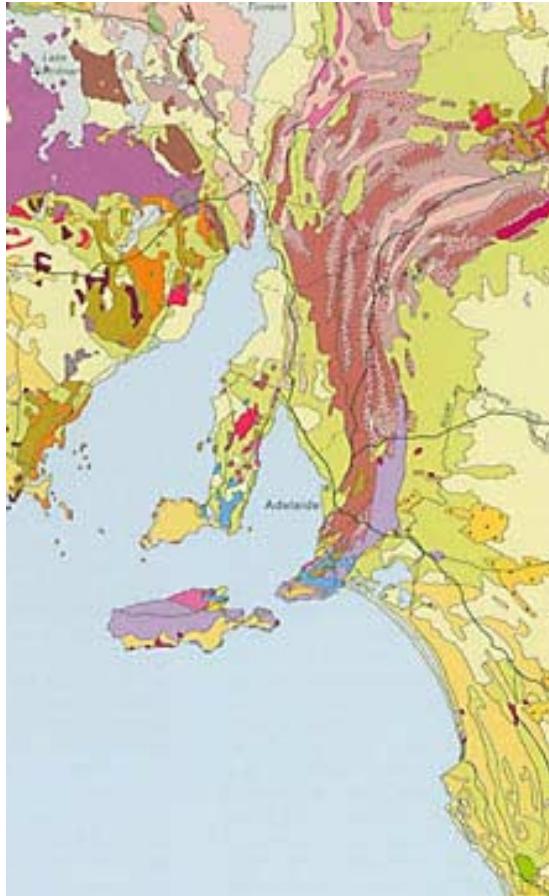
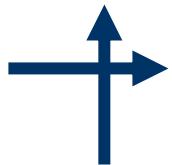
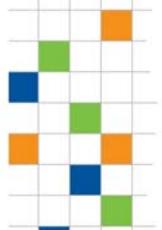
## Australia earthquake map ( $M > 4.0$ )

Queensland University Advanced Centre for Earthquake Studies  
**(QUAKES)**



June 06 / 9

# Earthquakes and Interacting Faults System

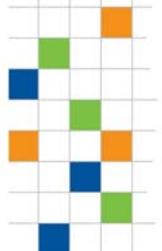


June 06 / 10

# Rupture Model

$$\rho \ddot{u}_i - (\sigma_{ij})_{,j} = 0$$

$$\sigma_{ij} = \lambda \cdot u_{k,k} \delta_{ij} + \mu \cdot (u_{i,j} + u_{j,i})$$



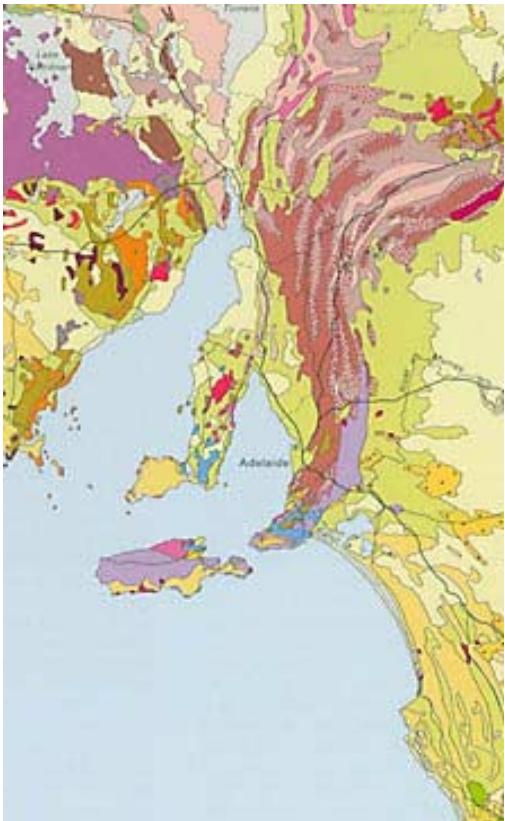
on faults:  $n_j \sigma_{ij} = f_n n_i + f_\tau \tau_i$

restoring force:  $f_n = \min(E_n[u]_j n_j, 0)$

friction force:  $f_\tau = \begin{cases} E_\tau \cdot \text{slip} + f_\tau^{\text{event}} & \text{for stick state} \\ \mu_d f_n & \text{for slip state} \end{cases}$

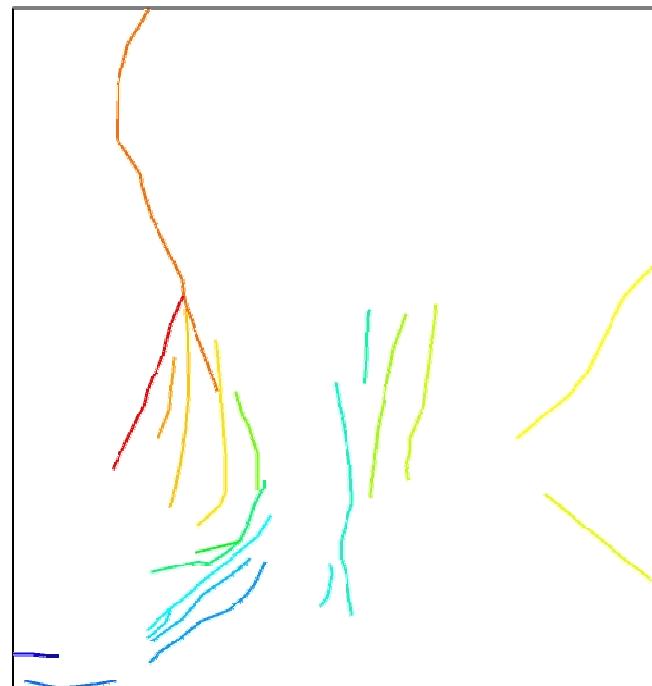
# Input Data

Rock types:  $\rho$ ,  $\lambda$ ,  $\mu$



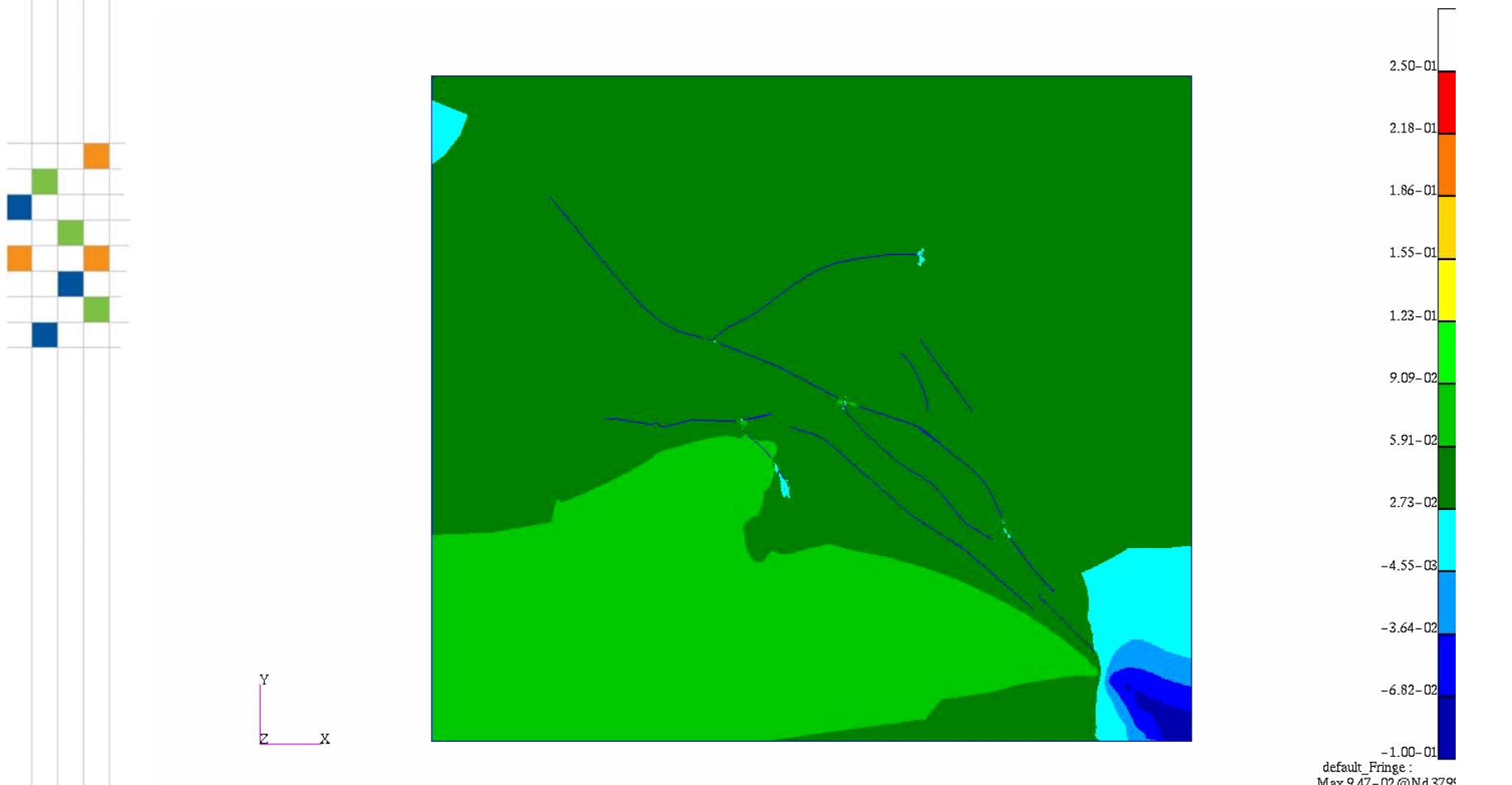
June 06 / 12

Fault Data:  $\mu_d$ ,  $E_n$ ,  $E_T$

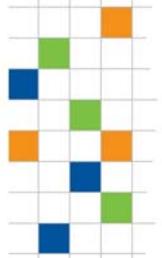


# Earthquake Cycles (Sth. CA, quasi static)

## Xing, Mora



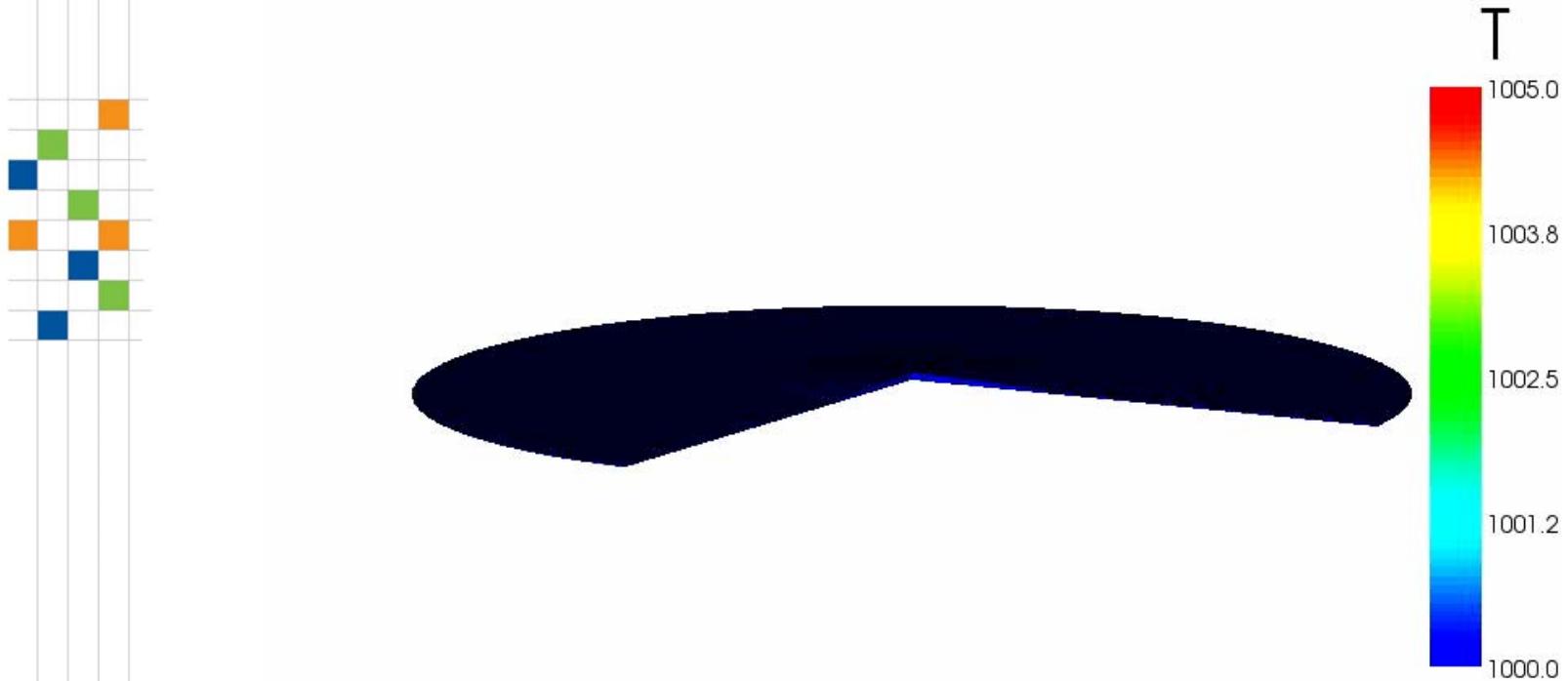
June 06 / 13



# Application: Volcanoes

# Volcanoes flow

frame: 0



June 06 / 15

# Characteristics

- Partial differential equation (PDE)
  - time-depended
  - non-linear
  - coupled
  - 3D FEM (or similar)
- Input Variables
  - constants or piecewise constant
  - output of other models
  - external sources

# Challenges (a selection)

- Large scale problems
- Code portability
- Coupling
- User interface
  - GUI
  - Grid services
- Experimental model development
- Reliability
- Simultaneous code base + application development

# Concept

Scope: GridSphere: APAC Grid

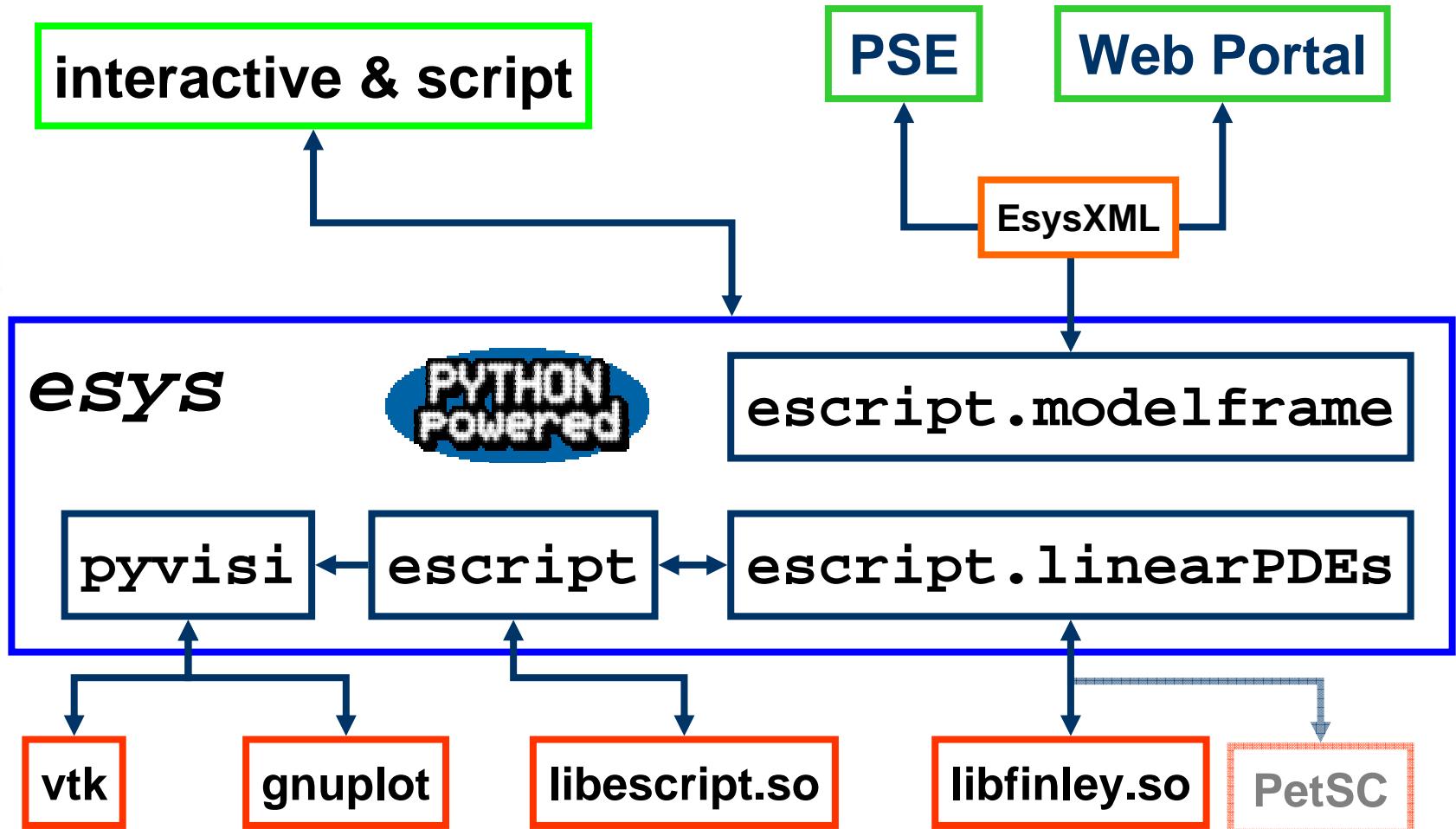
XML

Mathematics: Python: esys.escript

lib.so

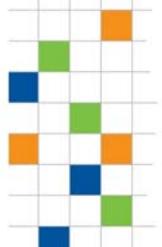
Numerics: C/C++: libfinley.so

# Architecture



# esys.escript

- Handles functions of spatial coordinates
  - on a Domain
  - in a FunctionSpace
    - defines sample points: nodes, cell centers, ...
- apply manipulations: +,-,\*, $\cos$ ,...
- data parallel: OpenMP + MPI
- R/W by numeric libraries/PDE solvers



# esys.escript (cont.)

- Transparent data representation
  - Constant
  - Expanded
  - Piecewise constant/tagged
- Invokes conversion
  - interpolation of sample point values
    - Handled by Domain(s)
  - change of representation

# escript.linearPDEs

- Interface to a general, linear PDE for  $u$
- coefficients are escript.Data objects
- Implemented by a solver library
  - through Domain interface
  - optimized for hardware

$$-(A_{klj}u_{i,j} + B_{kli}u_i)_{,l} + C_{kij}u_{i,j} + D_{ki}u_i = -(X_{k,l})_{,l} + Y_k$$

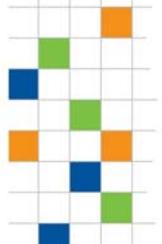
+ boundary conditions

# Finley Library

- Implements `escript.linearPDE`
- C-library to solve general, linear PDE
  - Finite Elements
  - Unstructured, isoparametric meshes
  - Contact Elements
- Parallelized
  - For ccNUMA architectures
  - MPI under construction

# Temperature Diffusion Model

$$\rho c_p T_{,t} - (\kappa T_{,j})_{,j} = Q$$



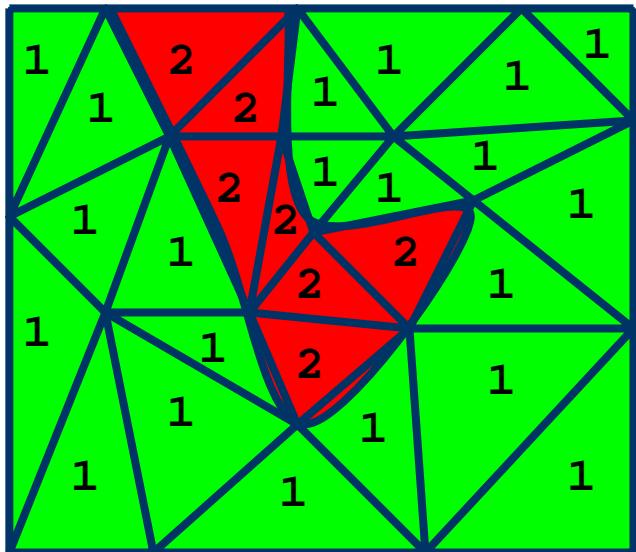
$$T_{,t} \rightarrow \frac{T^{(n)} - T^{(n-1)}}{dt}$$

$$\underbrace{- \underbrace{(\kappa \delta_{ij} T^{(n),i})_{,j}}_A + \underbrace{\frac{\rho c_p}{dt} T^{(n)}}_D}_{\text{LinearPDE class object}} = \underbrace{Q^{(n)} + \frac{\rho c_p}{dt} T^{(n-1)}}_Y$$

# Implementation

```
from esys.escript import *
from esys.finley import Brick
from esys.escript.linearPDEs import LinearPDE
mydomain=Brick()
kappa=1.
rhocp=Scalar(10., Function(mydomain))
mypde=LinearPDE(mydomain)
mypde.setValue(A=kappa*kronecker(mydomain), \
               D=rhocp/dt)
T=0.
t=0.
while t<t_end :
    mypde.setValue(Y=Q(t+dt)+rhocp/dt*T)
    T=mypde.getSolution()
    t+=dt
```

# Piecewise Constant Functions



```
rhoCP=Scalar(10, \
    what=Function(mydomain))
rhoCP.addTaggedValue(1,0.67)
rhoCP.addTaggedValue(2,5.68e3)
```

No changes on:

```
myPDE.setValue(Y=Q(t+dt)+rhoCP/dt*T)
```

# `esys.escript.modelframe`

## Objective:

- Reusability of models
- Accessibility of models

Provides a framework to

- set model parameters
- couple models
- building XML interfaces to models
- automated check pointing

# Implementation

- models are represented by objects

```
class Temperature(Model):
```

```
    <some definitions>
```

```
    my_model=Temperature()
```

- model parameters are object attributes

```
my_model.kappa=5.6
```

```
my_model.rhocp=Scalar(10,...)
```

```
my_model.Q=Link(other_model,"Q")
```

# Simple example

## Model library

```
class Temperature(Model):
    <temperature diffusion>
class Fluid(Model):
    <fluid flow depending on viscosity>
class MaterialTable(Model):
    <calculates viscosity from temperature>
```

```
tempmodel=Temperature()
tempmodel.density=8.9
flow=Fluid()
mat=MaterialTable()
mat.temperature=Link(tempmodel,"temperature")
flow.viscosity=Link(mat,"viscosity")
s=Simulation([tempmodel,mat,flow])
s.writeXML() # create ESySXML representation
s.run()
```

ESys Software Suite - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://jamboree.esscc.uq.edu.au/~doreenk/ igmt

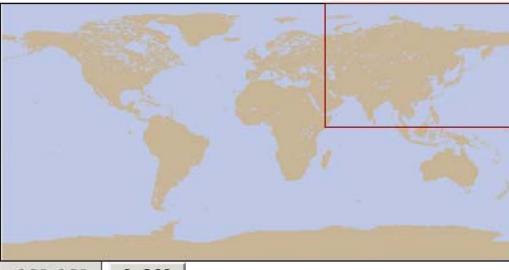
library CiteSeer ESSCC Twiki Python/XML Librar... nf.apac.edu.au M... APACgrid Twiki LibraryThing | Cat... AFT thingywhatsit

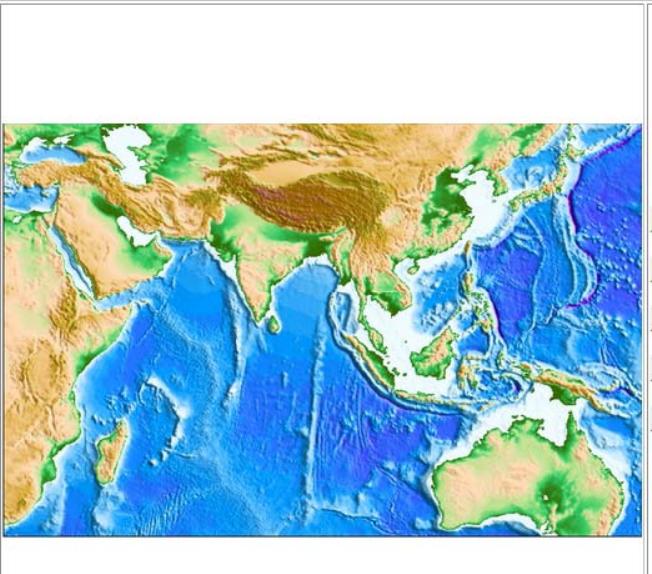
Australian Computational Earth Systems Simulator  
MAJOR NATIONAL RESEARCH FACILITY

## GMT and Tsunami Modelling Portal

west [47.2] east [180]  
south [4] north [90] draw square

Rivers  
 Cities  major  all  
 Volcanoes  show names  
 Earthquakes  show magnitudes  
 Hotspots  show names  
 Shorelines  
 State Boundaries  
 Plate Boundaries  
 WSM data  
colormap:  resolution(min): 10

  
-180..180 0..360



map me  
simulate  
help  
save grdfile  
save datafile

Designed by Doreen Kuehling

Powered by escript, python, triangle, GMT, vtk.

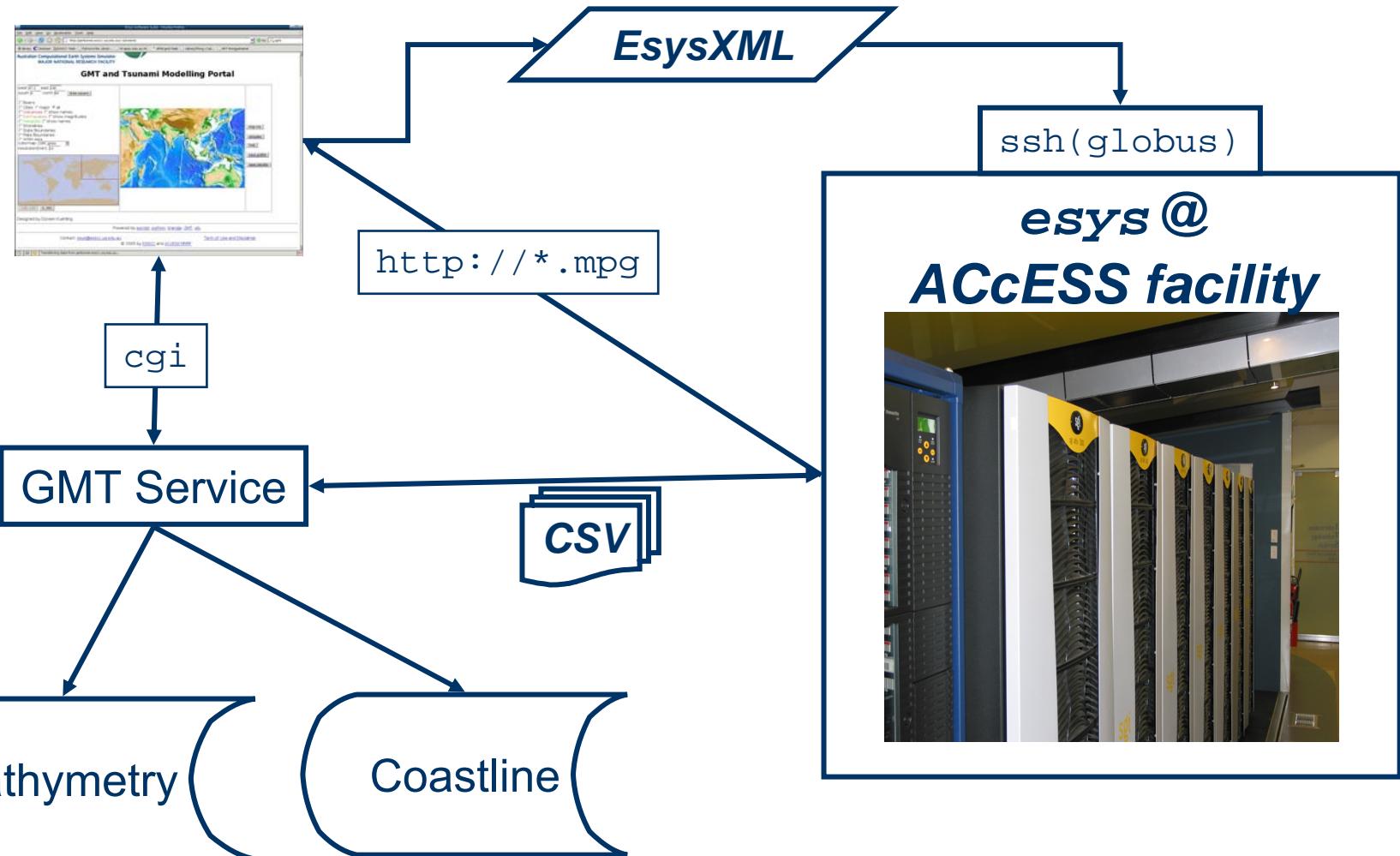
Contact: [esys@esscc.uq.edu.au](mailto:esys@esscc.uq.edu.au) © 2005 by ESSCC and ACCESS MNRF [Term of Use and Disclaimer](#)

Transferring data from jamboree.esscc.uq.edu.au...

June 06 / 30



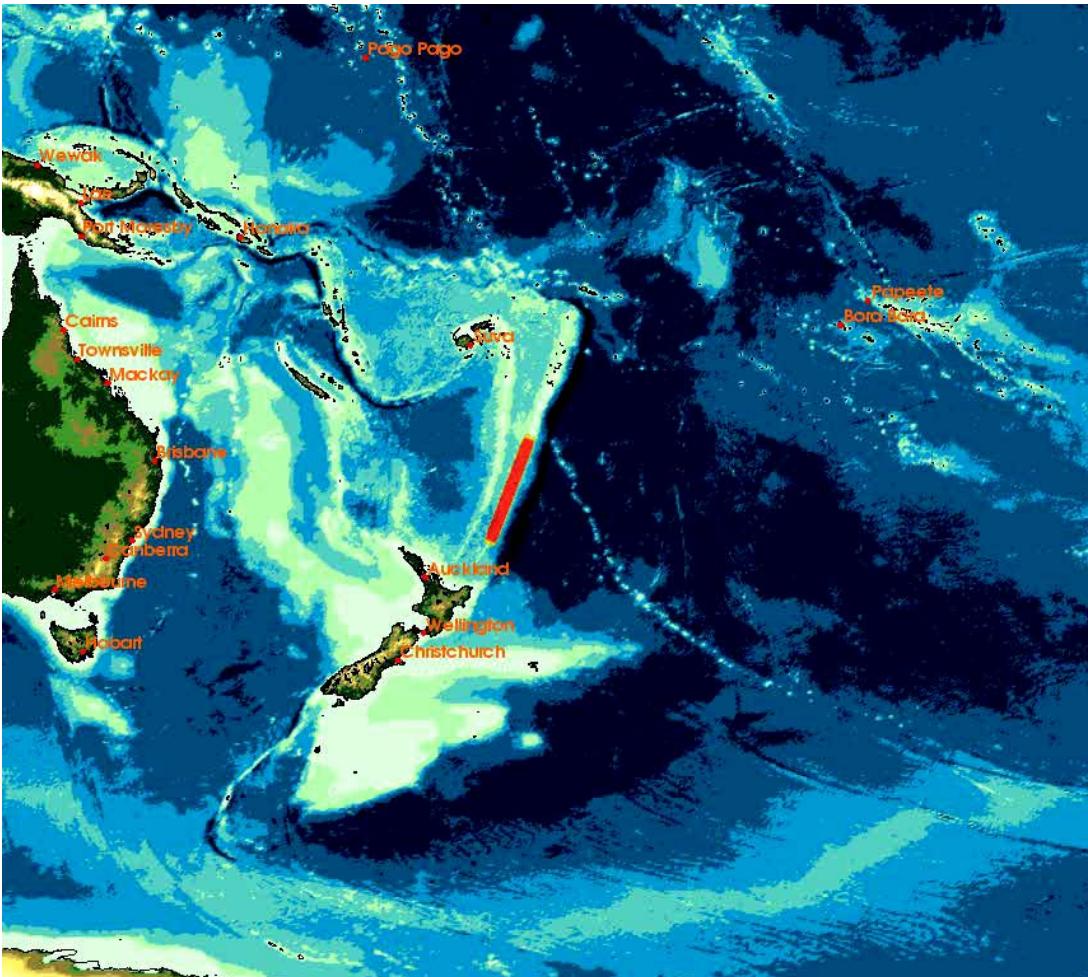
# Dataflow: Tsunami modelling portal



June 06 / 31

# Surf the Wave: Tsunamis

Weatherly, Gross, Cochrane



June 06 / 32

# Conclusive Remarks

- Rich test sets
- Performance issues:
  - implicit vs. explicit schemes
  - model complexity
  - Python polution
- Make escript GML compatible !!!
- Beta release available: [www.access.edu.au](http://www.access.edu.au)
  - MPI version end '06.