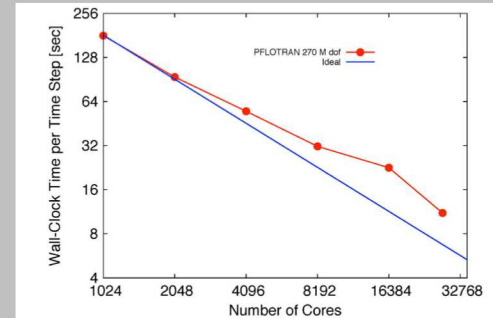
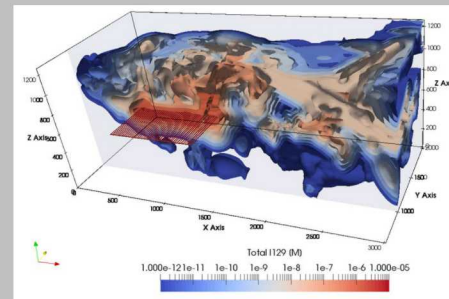


Exceptional service in the national interest



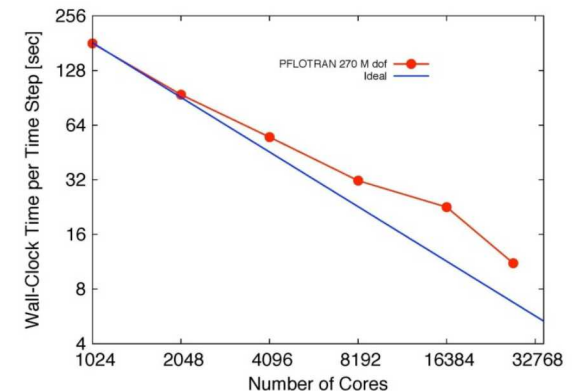
$$\begin{aligned}\frac{\partial m_a}{\partial t} &= -\nabla \cdot (\rho_l X_a^l \mathbf{q}_l + \rho_g X_a^g \mathbf{q}_g + \mathbf{J}_a^l + \mathbf{J}_a^g) + q_a^G, \\ \frac{\partial m_w}{\partial t} &= -\nabla \cdot (\rho_l X_w^l \mathbf{q}_l + \rho_g X_w^g \mathbf{q}_g + \mathbf{J}_w^l + \mathbf{J}_w^g) + q_w^G, \\ \frac{\partial e}{\partial t} &= -\nabla \cdot (\rho_l H_l \mathbf{q}_l + \rho_g H_g \mathbf{q}_g - \kappa_{\text{eff}} \nabla T) + q_e^G,\end{aligned}$$



PFLTRAN Overview

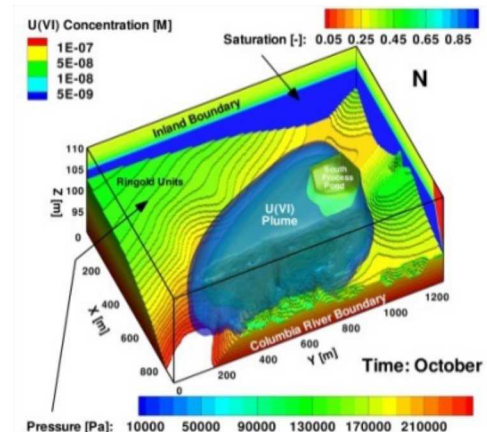
PFLOTRAN

- Petascale reactive multiphase flow and transport code
- Open source license (GNU LGPL 2.0)
- Object-oriented Fortran 2003/2008
 - Pointers to procedures
 - Classes (extendable derived types with member procedures)
- Founded upon well-known (supported) open source libraries
 - MPI, PETSc, HDF5, METIS/ParMETIS/CMAKE
- Demonstrated performance
 - Maximum # processes: 262,144 (Jaguar supercomputer)
 - Maximum problem size: 3.34 billion degrees of freedom
 - Scales well to over 10K cores



Application of PFLOTRAN

- Nuclear waste disposal
 - Waste Isolation Pilot Plant (WIPP) in Carlsbad, NM
 - US DOE NE Spent Fuel and Waste Science and Technology (SFWST)
 - SKB Forsmark Spent Fuel Nuclear Waste Repository (Sweden, Amphos²¹)
- Climate: coupled overland/groundwater flow; CLM
 - Next Generation Ecosystem Experiments (NGEE) Arctic
 - DOE Earth System Modeling (ESM) Program
- Biogeochemical transport modeling
 - U(VI) fate and transport at Hanford 300 Area
 - Hyporheic zone biogeochemical cycling
 - Columbia River, WA, USA
 - East River, CO, USA
- CO₂ sequestration
- Enhanced geothermal energy
- Radioisotope tracers
- Colloid-facilitated transport

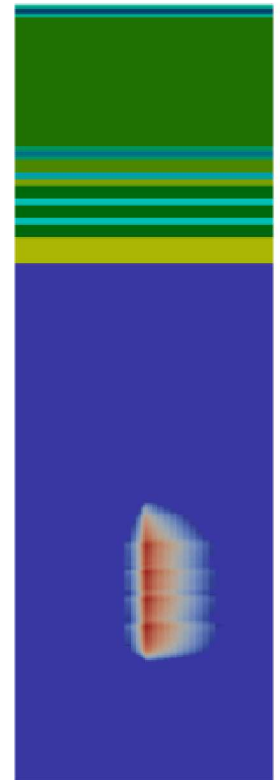


Hammond and Lichtner, WRR, 2010

Discretization and Numerical Methods

- Spatial discretization
 - Finite volume (2-point flux default)
 - Structured and unstructured grids
- Time discretization: fully-implicit backward Euler
- Nonlinear solver
 - Newton-Raphson
 - Line search/damping with custom convergence criteria
- Linear solver: direct (LU) or iterative (BiCGStab)
- Multi-physics coupling
 - Flow and transport/reaction: sequential
 - Transport and reaction: global implicit
 - Geomechanics and flow/transport: sequential

Deep Borehole
Waste Disposal

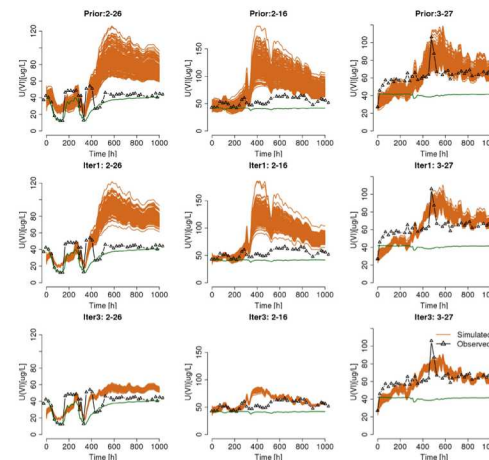


Emily Stein, SNL, 2015

PFLOTRAN Computing Capability

- High-Performance Computing (HPC)
 - Increasingly mechanistic process models
 - Highly-refined 3D discretizations
 - Massive probabilistic runs
- Open Source Collaboration
 - Leverages a diverse scientific community
 - Sharing among subject matter experts and stakeholders from labs/universities
- Modern Fortran (2003/2008)
 - Domain scientists remain engaged
 - Modular framework for customization
- Leverages Existing Capabilities
 - Meshing, visualization, HPC solvers, etc.
 - Configuration management, testing, and QA

Data Assimilation



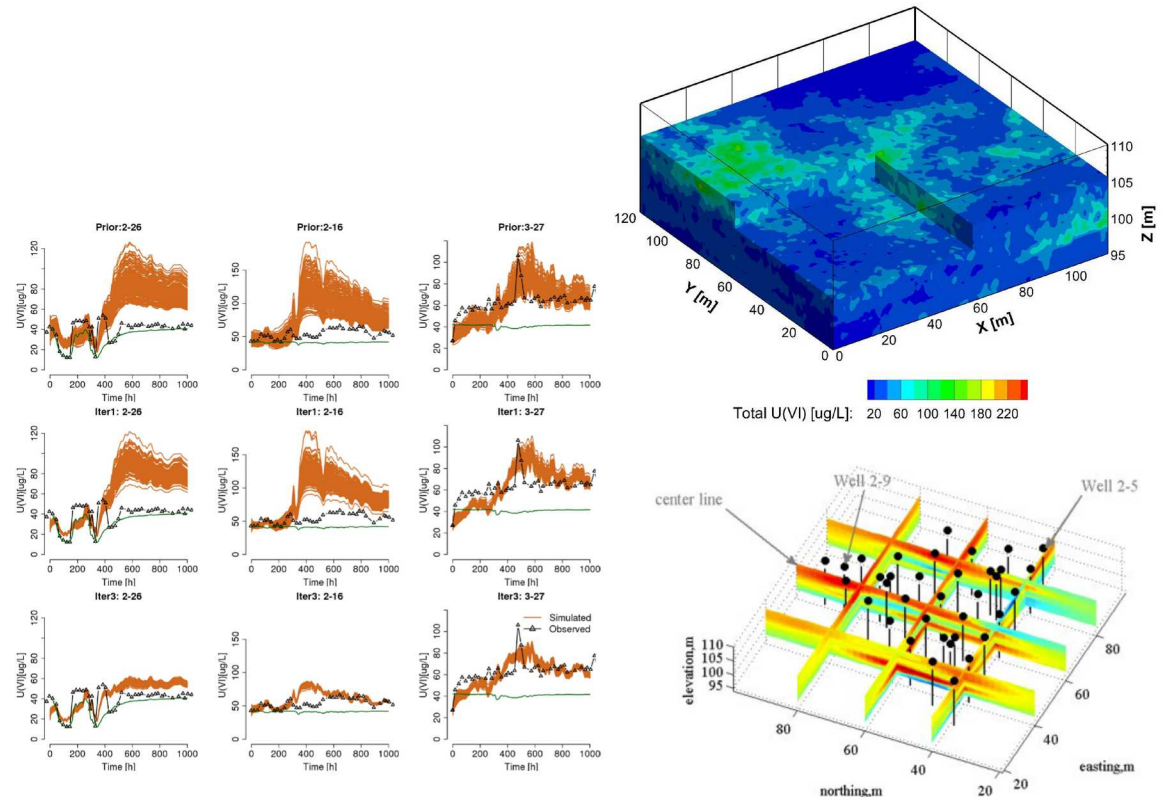
Xingyuan Chen, PNNL, 2011



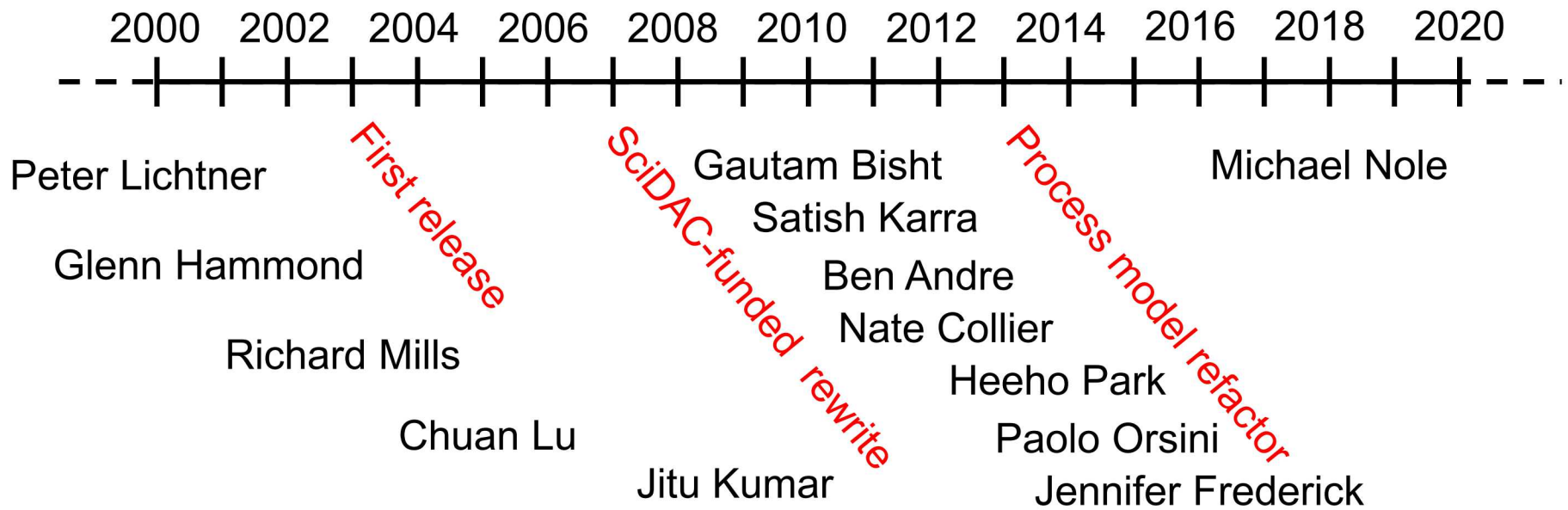
Benefits of Massively Parallel HPC

Data Assimilation at the Hanford 300 Area (Xingyuan Chen, 2011)

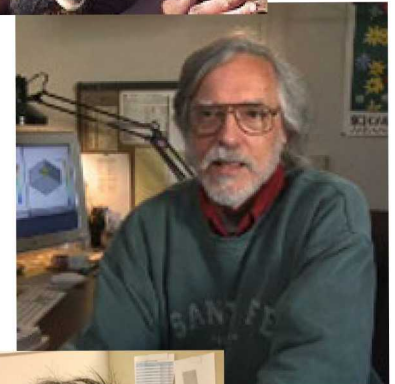
- Problem domain:
 - $120 \times 120 \times 15\text{m}$
 - $\Delta x, \Delta y, \Delta z = 1, 1, 0.5\text{ m}$
 - 432K grid cells
 - 15 chemical species
 - 6.48M dofs total
- 1-2 month simulation:
 - $\Delta t = 1\text{ hour}$
- Computing, e.g.
 - 128 cores
(single realization)
 - 64,000 cores
(500 realizations)
 - 1 hour wallclock runtime
 - ~7 cpu years



PFLOTRAN Development Timeline



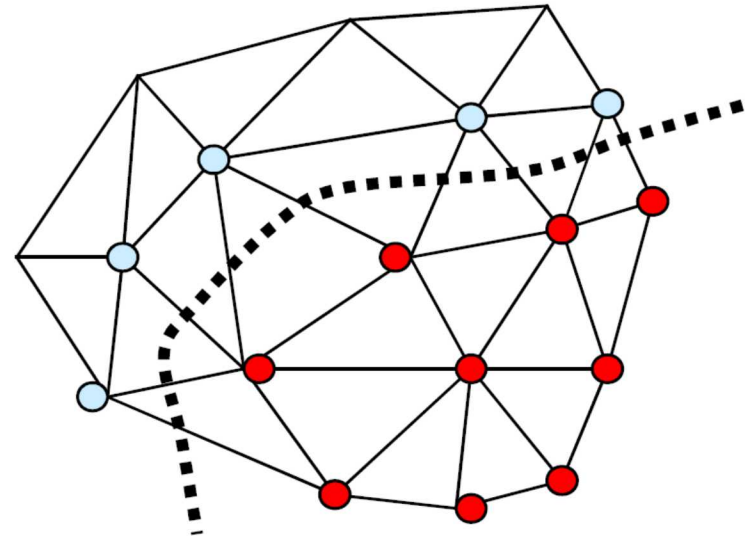
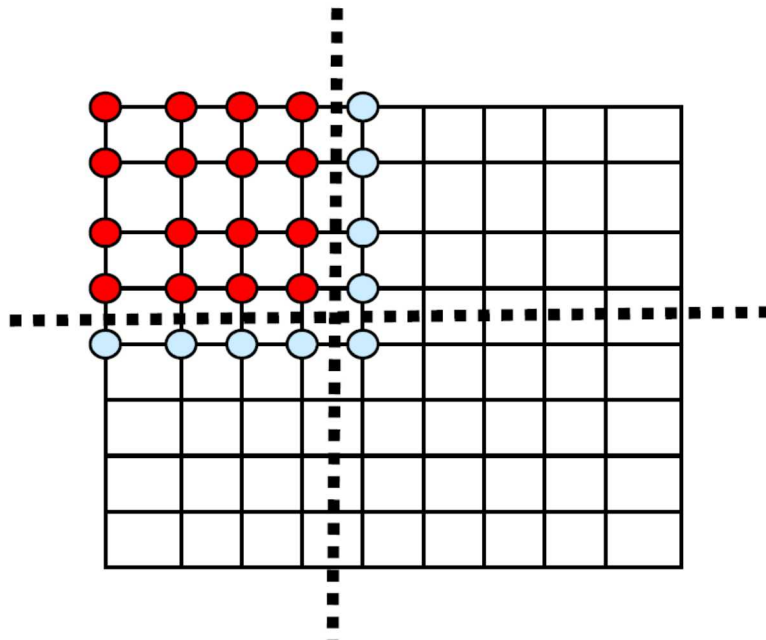
PFLOTRAN Developers



Approach to Parallelization

Domain Decomposition

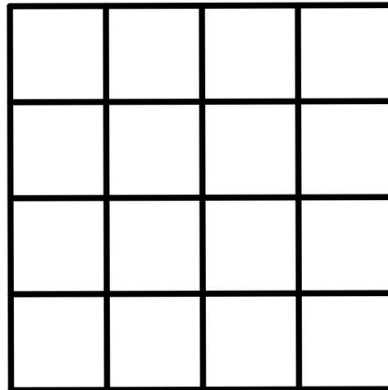
● Local node ○ Ghost node



Ghost node information used only in flux calculations.

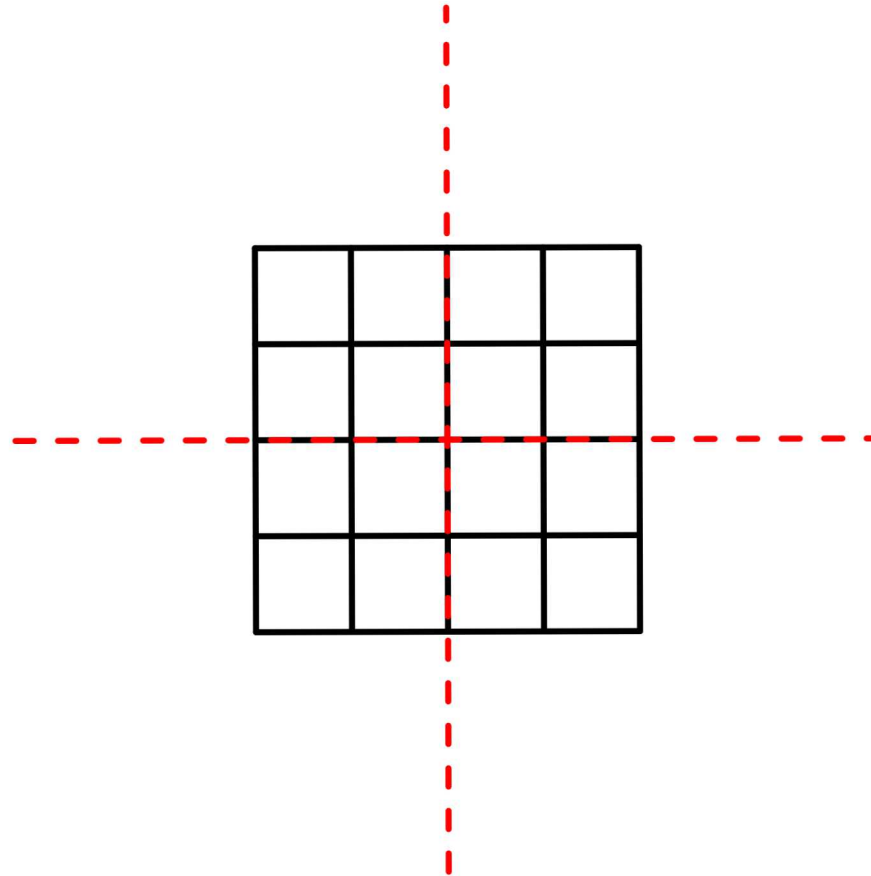
Domain Decomposition

Structured Grid Example



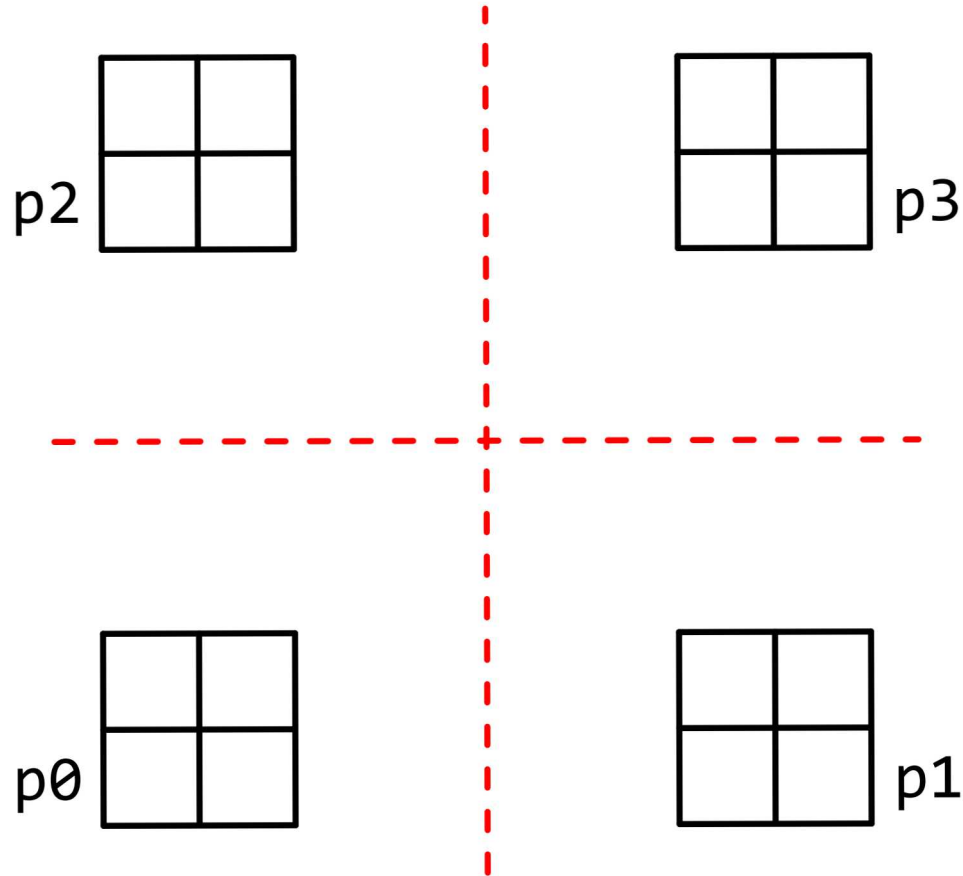
Domain Decomposition

Structured Grid Example



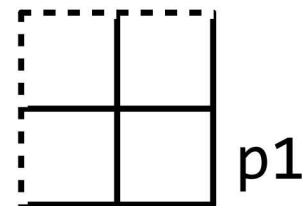
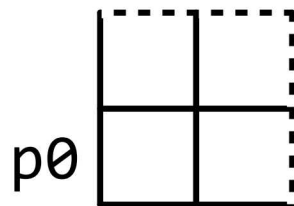
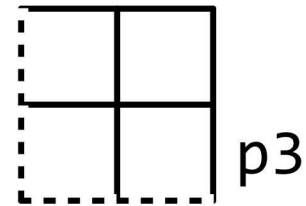
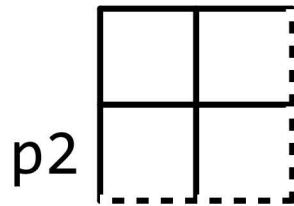
Domain Decomposition

Structured Grid Example



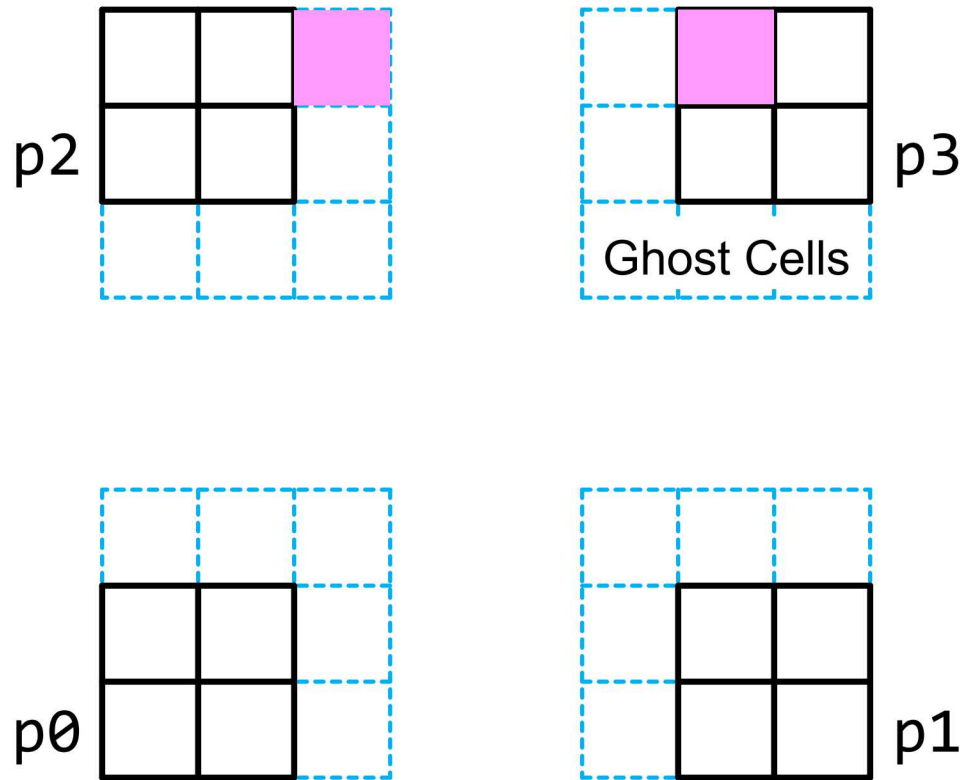
Domain Decomposition

Structured Grid Example



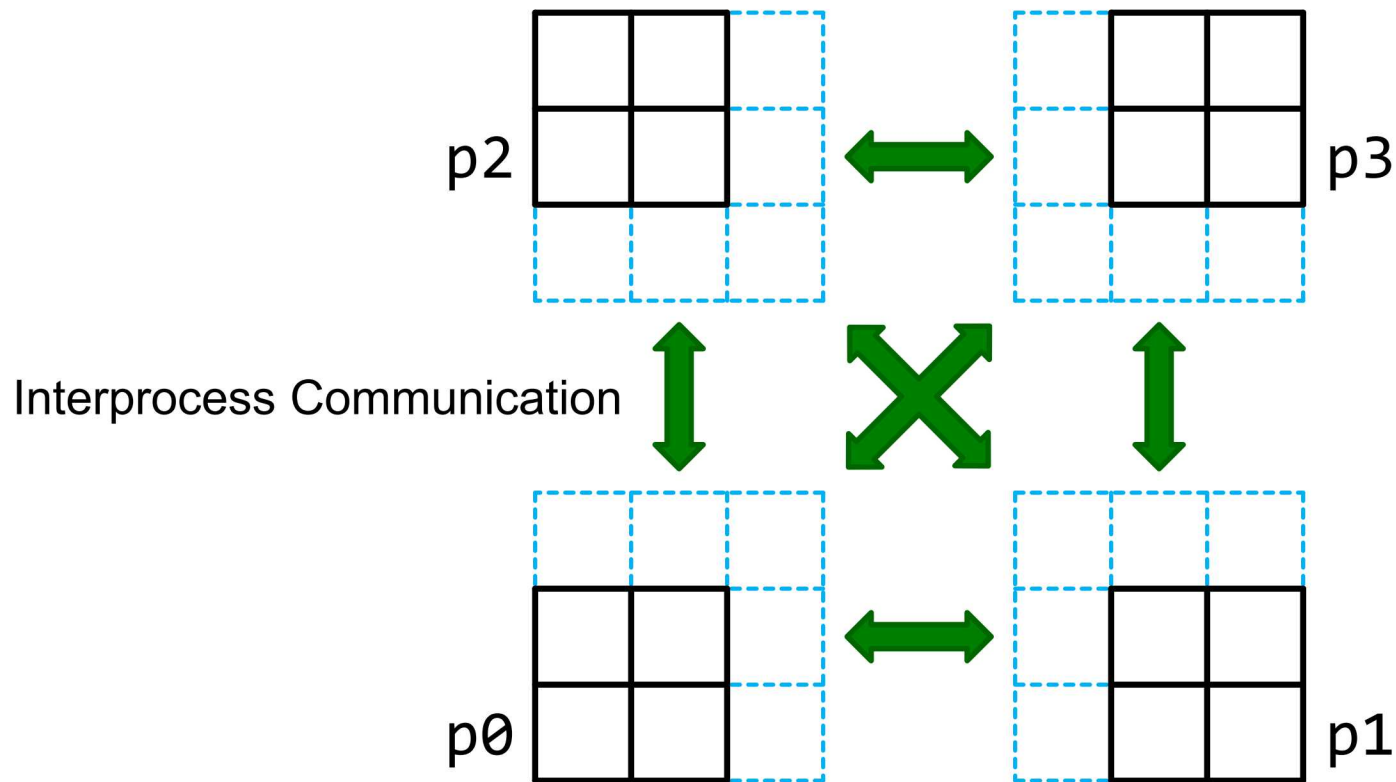
Domain Decomposition

Structured Grid Example



Domain Decomposition

Structured Grid Example



Installing PFLOTRAN

- `git clone https://bitbucket.org/petsc/petsc`
- `cd petsc`
- `git checkout v3.10.2`
- `PETSC_DIR=`pwd` ; PETSC_ARCH=gnu`
- `./configure --download-mpich=yes --download-hdf5=yes --download-parmetis=yes --download-metis=yes --download-fblaslapack=yes`
- `make all`
- `cd path/to/pflotran/installation/directory`
- `git clone https://bitbucket.org/pflotran/pflotran`
- `cd pflotran/src/pflotran`
- `make pflotran`

Assumes Git, GNU compilers, and Python 2.7+ are installed

Third-Party Libraries

- `git clone https://bitbucket.org/petsc/petsc`
- `cd petsc`
- `git checkout v3.10.2`
- `PETSC_DIR=`pwd` ; PETSC_ARCH=gnu`
- `./configure --download-mpich=yes --download-hdf5=yes --download-parmetis=yes --download-metis=yes --download-fblaslapack=yes`
- `make all`
- `cd path/to/pflotran/installation/directory`
- `git clone https://bitbucket.org/pflotran/pflotran`
- `cd pflotran/src/pflotran`
- `make pflotran`

Executing PFLOTTRAN

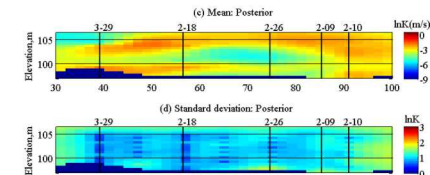
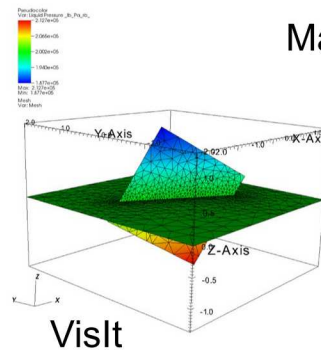
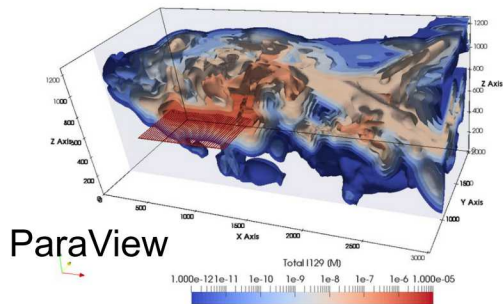
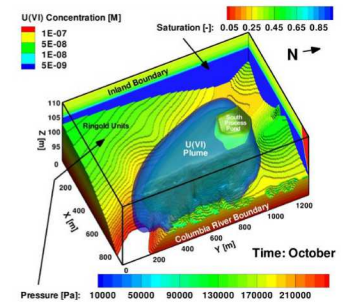
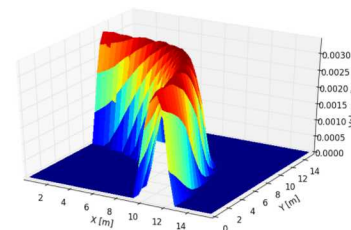
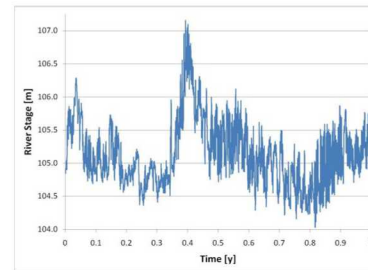
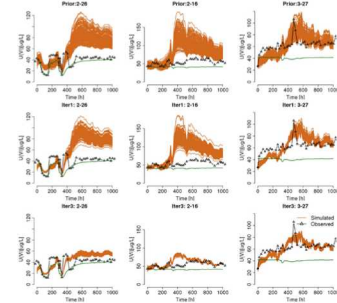
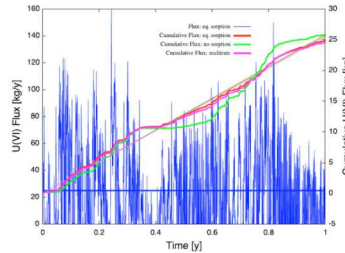
- `pflotran`
 - Serial execution
- `mpirun -n # pflotran`
 - Assumes input file is named `pflotran.in`
- `mpirun -n # pflotran -input_prefix <string1> -output_prefix <string2>`
 - Input filename is named `<string1>.in`
 - Output filenames are named `<string2>.*`
 - `<string2> = <string1>` if `-output_prefix` missing.
- `mpirun -n # pflotran -pflotranin <filename>`
 - Sets `input_prefix` to `<filename>` with `.in` removed

Executing PFLOTTRAN (Advanced)

- `mpirun -n # pflotran -stochastic
-num_realizations # -num_groups #`
 - Run multiple simulations (realizations) at once
 - Samples datasets based on realization #
 - # concurrent realizations = `num_groups`
 - # processes per simulation = `n/num_groups`
 - # realizations per group = `num_realizations/num_groups`
- `mpirun -n # pflotran -realization_id #`
 - Run an individual realization

PFLOTRAN Output Formats

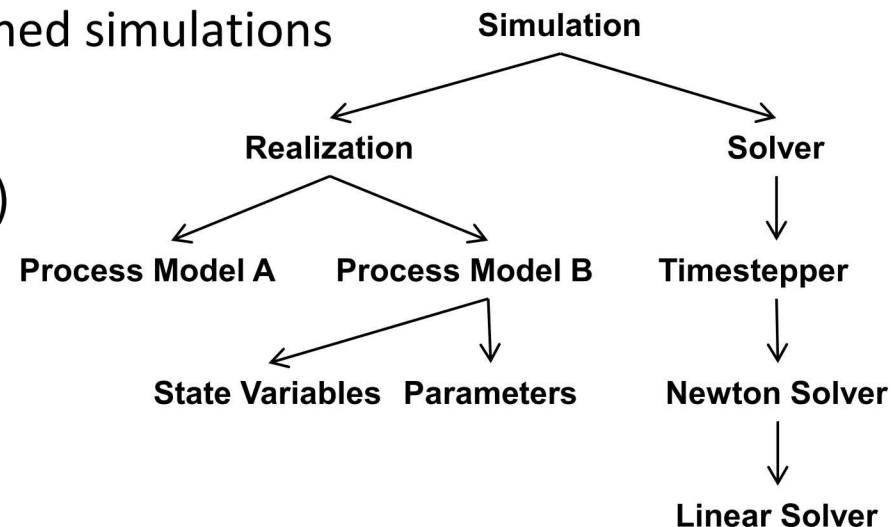
- Excel: *.tec [POINT]
- gnuplot: *.tec [POINT]
- MATLAB: *.tec [POINT], *.h5
- Matplotlib: *.tec [POINT]
- Tecplot : *.tec [POINT, BLOCK, FEBRICK]
- ParaView: *.h5, *.xmf, *.vtk
- R *.tec [POINT], *.h5
- VisIt *.h5, *.xmf, *.vtk



Why Object-Oriented Fortran 2003/2008?

- Why Fortran?
 - Experienced domain scientists remain engaged
 - Commonality among all domain scientists
- Why object-oriented?
 - Modular data structures
 - Eases code development and debugging – data locality
 - Nesting of processes and data
 - Tree structure enables self-contained simulations

- Why Fortran 2003/2008?
 - Classes (extendable derived types)
 - Member functions
 - Inheritance
 - Pointers to procedures
 - E.g. swapping equations of state



Object Oriented Fortran

Fortran 77

```
subroutine X(a,b,c,d,e,f,...)
```

```
...
```

```
common/array/a(ncomp,ncell)
```

```
do icell = 1, ncell  
  do icomp = 1, ncomp  
    a(icomp,icell) = ...  
  enddo  
enddo
```

OO Fortran 90

```
subroutine X(realization)
```

```
...
```

```
grid => realization%patch%grid  
reaction => realization%reaction  
cells => realization%patch...%cells
```

```
do icell = 1, grid%ncell  
  do icomp = 1, reaction%ncomp  
    cells(icell)%conc(icomp) = ...  
  enddo  
enddo
```

Fortran 2003/2008

Fortran 90

```
select case(eos_type)
  case(WATER)
    call EvaluateWater(p,t)
  case(AIR)
    call EvaluateAir(p,t)
  case(CO2)
    call EvaluateCO2(p,t)
  case(CH4)
    call EvaluateCH4(p,t)
end select
```

Fortran 2003/2008

```
type, extends(eos_base) :: eos_CO2
  ...
contains
  procedure :: Evaluate => EvaluateCO2
end type eos_CO2

class(eos_CO2) :: eos

call eos%Evaluate(p,t)
```

eos = equation of state

PFLOTRAN Flow Modes

- Covered in shortcourse
 - RICHARDS: variably-saturated water
 - TH: variably-saturated water-energy
 - GENERAL: multiphase air-water-energy
- Not covered (multiphase)
 - MPHASE: supercritical CO₂-water-energy
 - WIPP_FLOW: immiscible air-water
 - FLASH2: supercritical CO₂-water-energy (experimental)
 - IMMIS: immiscible air-water (experimental)
 - MISCIBLE: X-water-energy (experimental)
 - TOIL_IMS: oil-water-energy (experimental)

Governing Equations

General Mode (air-water-energy)

Water Mass

$$\frac{\partial \phi (s_l \rho_l X_w^l + s_g \rho_g X_w^g)}{\partial t} = -\nabla \cdot (\rho_l X_w^l \mathbf{q}_l + \rho_g X_w^g \mathbf{q}_g + \mathbf{J}_w^l + \mathbf{J}_w^g) + q_w$$

Air Mass

$$\frac{\partial \phi (s_l \rho_l X_a^l + s_g \rho_g X_a^g)}{\partial t} = -\nabla \cdot (\rho_l X_a^l \mathbf{q}_l + \rho_g X_a^g \mathbf{q}_g + \mathbf{J}_a^l + \mathbf{J}_a^g) + q_a$$

Energy

$$\frac{\partial \phi (s_l \rho_l U_l + s_g \rho_g U_g) + (1 - \phi) C_p^{\text{rock}} \rho_{\text{rock}} T}{\partial t} = -\nabla \cdot (\rho_l H_l \mathbf{q}_l + \rho_g H_g \mathbf{q}_g - \kappa_{\text{eff}} \nabla T) + q_e$$

ϕ = effective porosity [-]

s_l = liquid saturation [-]

s_g = gas saturation [-]

ρ_l = liquid phase density [kmol/m³]

ρ_g = gas phase density [kmol/m³]

X_w^l = mole fraction of water in the liquid phase [-]

X_w^g = mole fraction of water in gas phase [-]

X_a^l = mole fraction of air in the liquid phase [-]

X_a^g = mole fraction of air in gas phase [-]

$\mathbf{q}_l$ = liquid phase Darcy flux [m/sec]

$\mathbf{q}_g$ = gas phase Darcy flux [m/sec]

$\mathbf{J}_w^l$ = water diffusive flux in liquid phase [kmol/m²/sec]

$\mathbf{J}_a^l$ = air diffusive flux in liquid phase [kmol/m²/sec]

$\mathbf{J}_w^g$ = water vapor diffusive flux in gas phase [kmol/m²/sec]

$\mathbf{J}_a^g$ = air diffusive flux in gas phase [kmol/m²/sec]

q_w = water source/sink [kmol/sec]

q_a = air source/sink [kmol/sec]

q_e = energy source/sink [MJ/sec]

U_l = liquid phase internal energy [MJ/kmol]

U_g = gas phase internal energy [MJ/kmol]

H_l = liquid phase enthalpy [MJ/kmol]

H_g = gas phase enthalpy [MJ/kmol]

C_p^{rock} = rock heat capacity [MJ/kg rock-K]

ρ_{rock} = rock particle density [kg/m³ rock]

T = temperature [C]

κ_{eff} = effective thermal conductivity [W/K-m]

Governing Equations

Darcy Flux

$$\mathbf{q}_{\alpha} = -\frac{k k_{\alpha}}{\mu_{\alpha}} \nabla (p_{\alpha} - \gamma_{\alpha} g z), \quad (\alpha = l, g)$$

Air Diffusion in Liquid Phase

$$\mathbf{J}_a^l = -\tau \phi s_l D_l \rho_l \nabla X_a^l$$

Air Diffusion in Gas Phase

$$\mathbf{J}_a^g = -\tau \phi s_g D_g^0 \left(\frac{T}{T_K} \right)^{\theta} \frac{p_0}{p_g} \rho_g \nabla X_a^g$$

$\mathbf{q}_{\alpha}$ = Darcy flux for phase α [m/s]

k = intrinsic permeability [m^2]

k_{α} = relative permeability for phase α [-]

μ_{α} = viscosity for phase α [Pa-s]

p_{α} = pressure for phase α [Pa]

γ_{α} = density for phase α [kg/m^3]

g = gravity [m/s^2]

z = elevation [m]

$\mathbf{J}_a^l$ = diffusive flux of air in liquid phase [$\text{kmol}/\text{m}^2\text{-s}$]

τ = tortuosity [-]

D_l = aqueous diffusivity [m^2/s]

$\mathbf{J}_a^g$ = diffusive flux of air in gas phase [$\text{kmol}/\text{m}^2\text{-s}$]

D_g^0 = gas diffusivity [m^2/s]

p_0 = reference pressure [Pa]

Primary Variables

General Mode

Thermodynamic State of Fluid	Primary Variables		
	X_1	X_2	X_3
Two-Phase	p_g	s_g	T
	p_g	s_g	p_a
Liquid	p_l	X_a^l	T
Gas	p_g	p_a	T

Governing Equations

TH Mode (water-energy)

Water Mass

$$\frac{\partial \phi (s_l \rho_l)}{\partial t} = -\nabla \cdot (\rho_l \mathbf{q}_l) + q_w$$

Energy

$$\frac{\partial \phi (s_l \rho_l U_l) + (1 - \phi) C_p^{\text{rock}} \rho_{\text{rock}} T}{\partial t} = -\nabla \cdot (\rho_l H_l \mathbf{q}_l - \kappa_{\text{eff}} \nabla T) + q_e$$

Governing Equations

RICHARDS Mode (water)

Water Mass

$$\frac{\partial \phi(s_l \rho_l)}{\partial t} = -\nabla \cdot (\rho_l \mathbf{q}_l) + q_w$$

Constitutive Relations

- Characteristic curves
 - Capillary pressure / saturation functions
 - E.g. Brooks Corey, van Genutchen
 - Relative permeability functions
 - Burdine, Mualem
- Equations of state
 - Water/steam/gas density/enthalpy/viscosity
- Solubility of gas in liquid
 - $f(K_H)$

PFLOTRAN Reactive Transport

- Transport
 - Multicomponent
 - Mobile/immobile primary species
 - Advection (upwinding)
 - Hydrodynamic dispersion
- Reaction
 - Aqueous speciation
 - Ion activity models
 - General ($A + B \leftrightarrow C$)
 - N^{th} order kinetics
 - Reversible
 - Mineral precipitation-dissolution
 - Prefactors
- Microbiological
 - Michaelis–Menten kinetics
 - Biomass
 - Inhibition
- Radioactive decay with daughter products
- Sorption
 - Isotherm-based: linear, Langmuir, Freundlich
 - Ion exchange
 - Surface complexation
 - Equilibrium
 - Kinetic / multirate kinetic
- Reaction Sandbox

Governing Equations

$$\frac{\partial}{\partial t}(\phi s \Psi_j + \Psi_{S,j}) + \nabla \cdot (\mathbf{q} - \phi s \mathbf{D} \nabla) \Psi_j = Q_j - \sum_r \nu_{jr} I_r$$

$$\Psi_j = C_j + \sum_{i=1}^{N_X} \nu_{ji} \frac{K_i}{\gamma_i} \prod_{j'=1}^{N_c} (\gamma_{j'} C_{j'})^{\nu_{j'i}}$$

ϕ = porosity [-]

s = liquid saturation [-]

Ψ_j = total component concentration of species j [mol/L]

$\Psi_{S,j}$ = total sorbed concentration of species j [mol/m³]

$\mathbf{q}$ = liquid Darcy velocity [m/s]

$\mathbf{D}$ = hydrodynamic dispersion coefficient [m²/s]

Q_j = source/sink for species j [mol/s]

ν_{jr} = stoichiometry of species j in kinetic reaction r [-]

I_r = kinetic rate for reaction r [mol/s]

C_j = free ion concentration for species j [mol/kg water]

N_X = # aqueous complexes [-]

ν_{ji} = stoichiometry of species j in complex i [-]

K_i = equilibrium constant for complex i

γ_i = activity coefficient for complex i [-]

N_c = # primary aqueous species [-]

Newton-Raphson with Log Formulation

$$\mathbf{J}_{ij} = \frac{\partial \mathbf{f}_i(\mathbf{c}^p)}{\partial \ln c_j^p} = c_j^p \frac{\partial \mathbf{f}_i(\mathbf{c}^p)}{\partial c_j^p} \quad \text{for all } i, j$$

$$\mathbf{J} \delta \ln \mathbf{c}^p = -\mathbf{f}(\mathbf{c}^p)$$

$$\mathbf{c}^{p+1} = \mathbf{c}^p \exp(\delta \ln \mathbf{c}^p)$$

Input Deck

CHEMISTRY

...

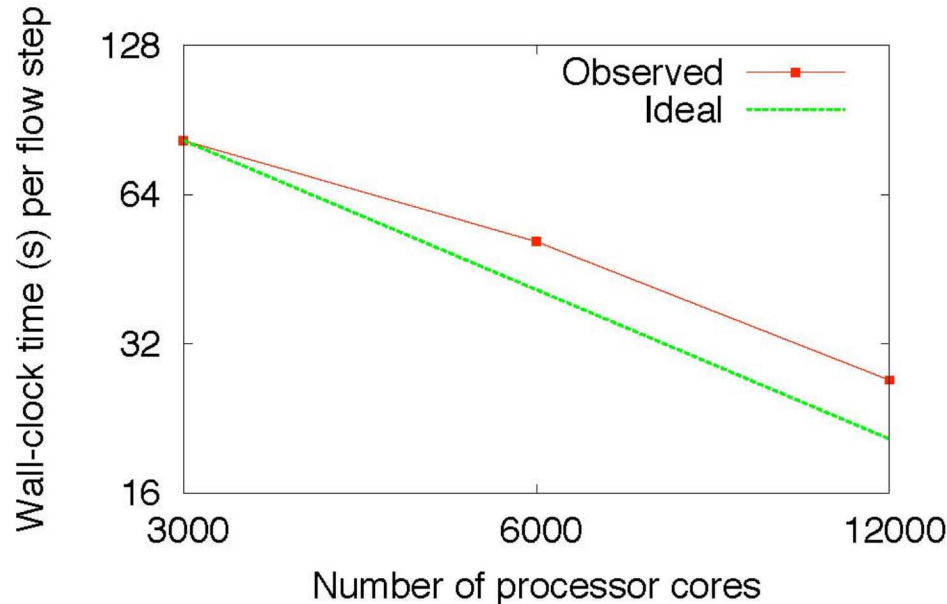
LOG_FORMULATION

...

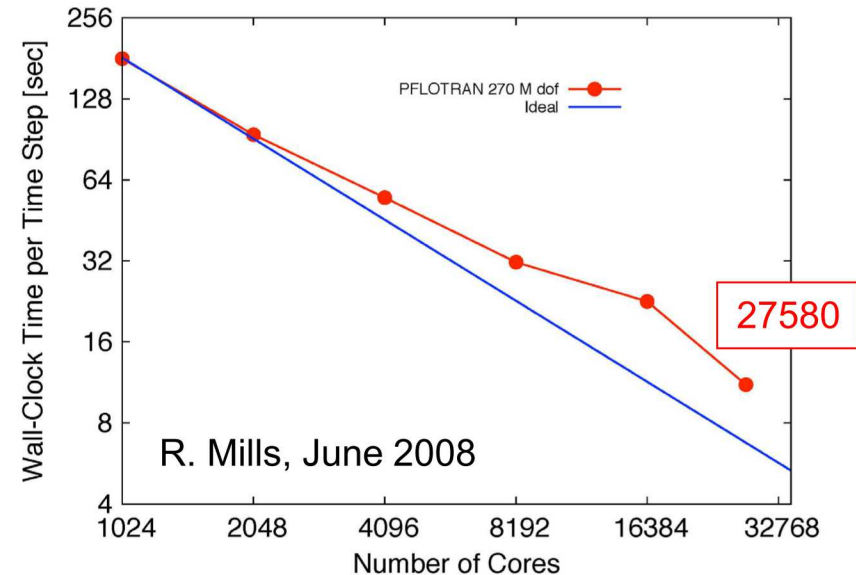
END

PFLOTRAN Parallel Performance

500M dof



270M dof



One-billion node performance on Jaguar

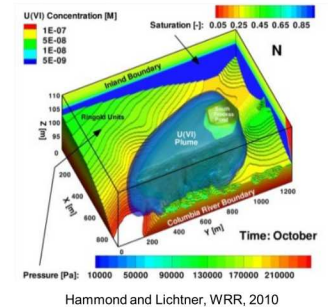
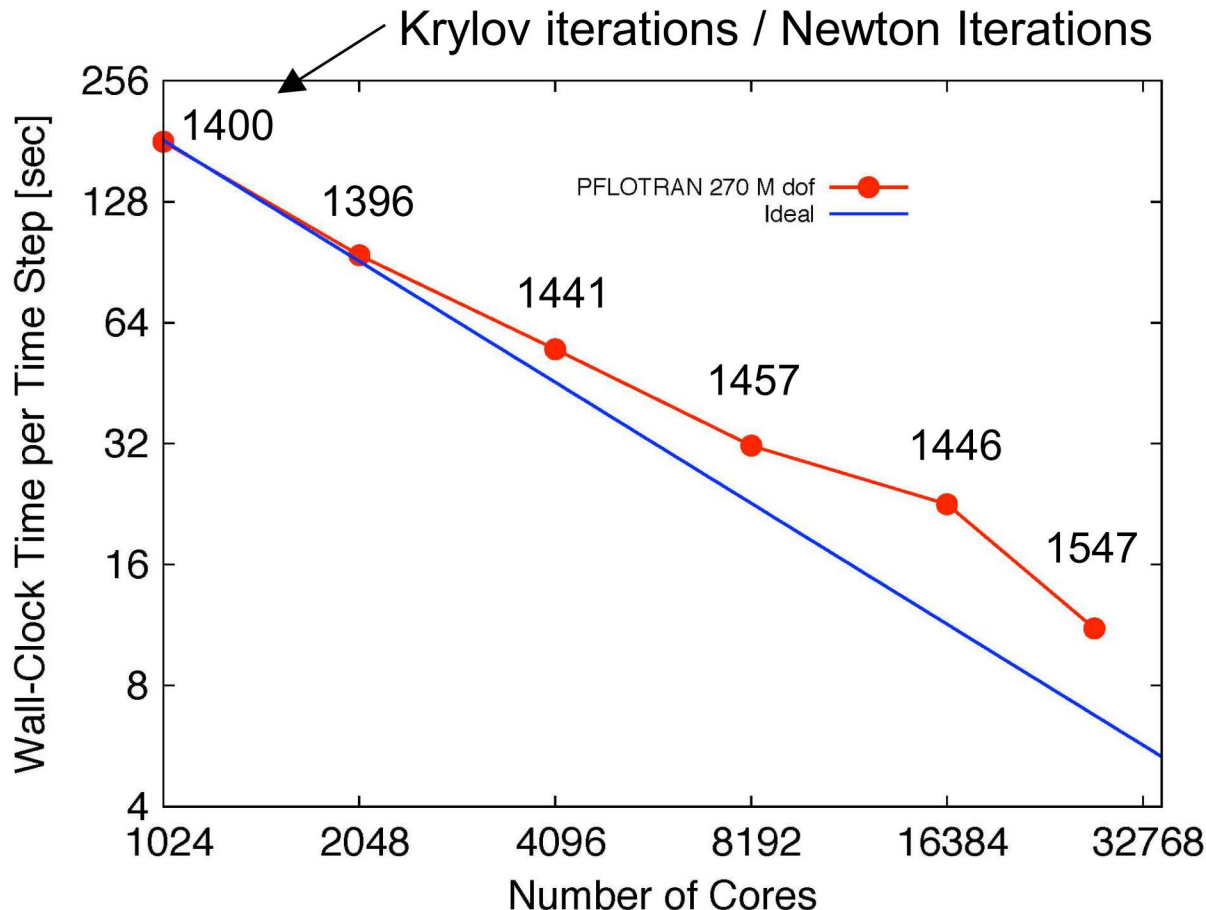
1B dof
proof of concept
1 time step

# procs	newton its	bcgs its	time(sec)
1024	16	1697	1143
2048	16	1626	697
4096	16	1720	325

87.9%
efficiency

PFLOTRAN Parallel Performance

Hanford 300 Area



Overall 60% efficient
Science

(1-2% of overall time)

- 75% efficient
(wall clock)

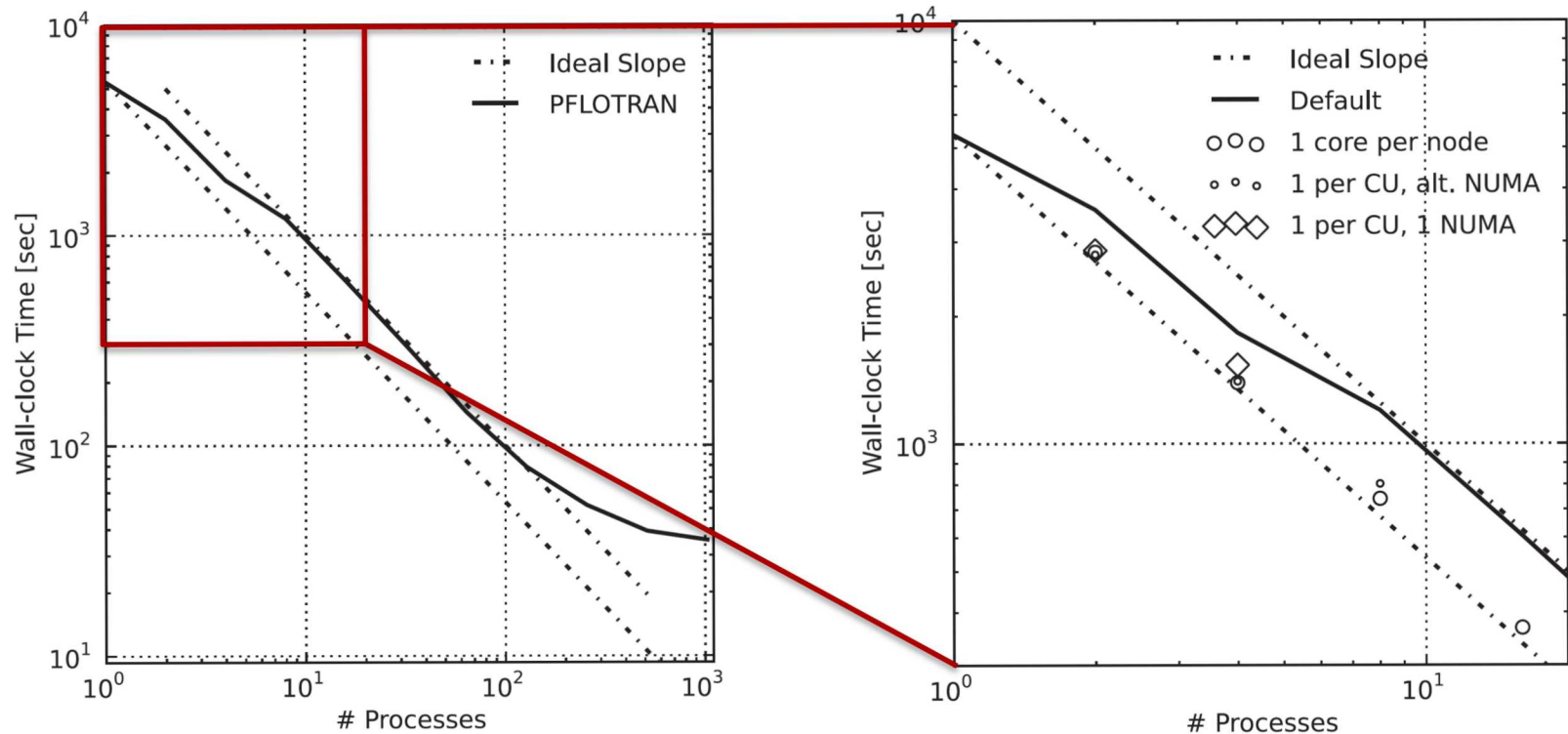
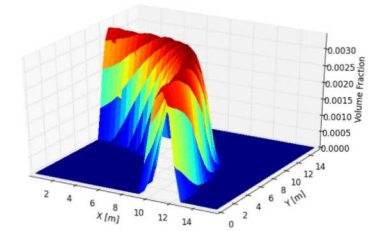
Solvers

(96% of overall time)

- 90% efficient
(iteration count)
- 64% efficient
(wall clock)

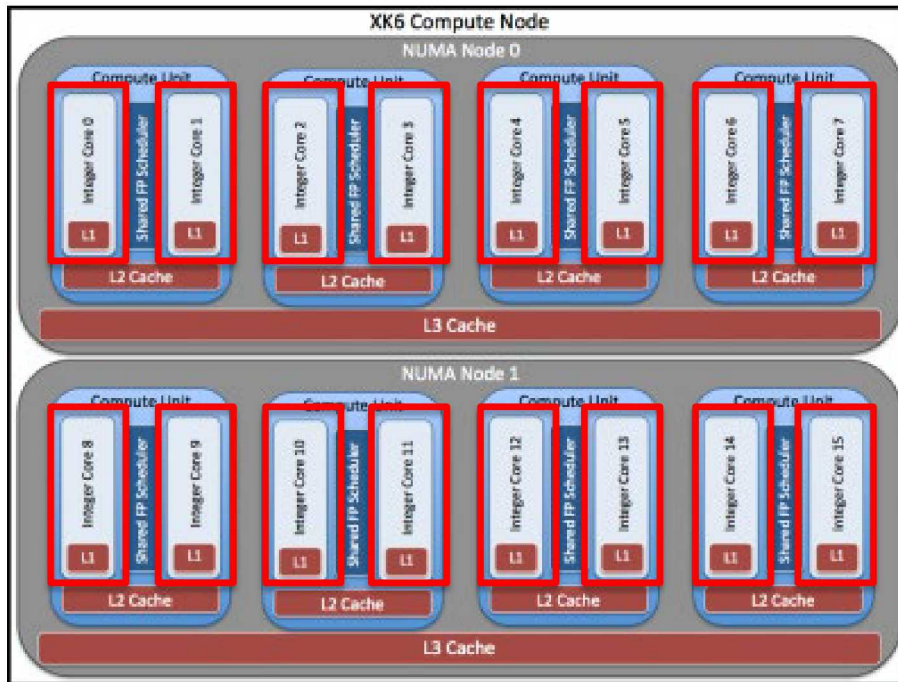
PFLOTRAN Single Node Performance

Copper Leaching

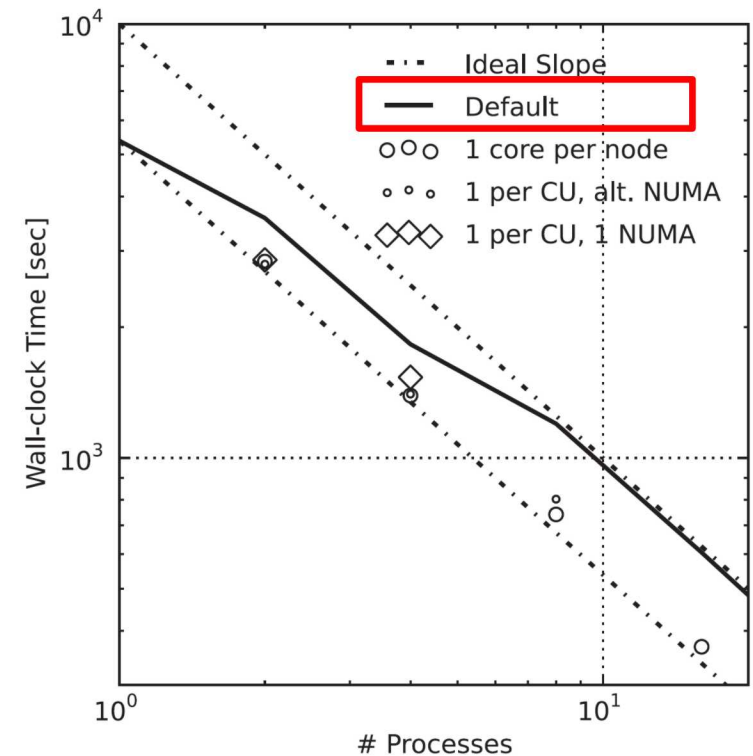
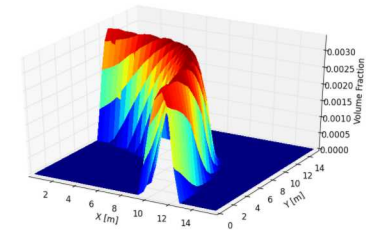


PFLOTRAN Single Node Performance

XK6 Compute Node with 2 NUMA Nodes



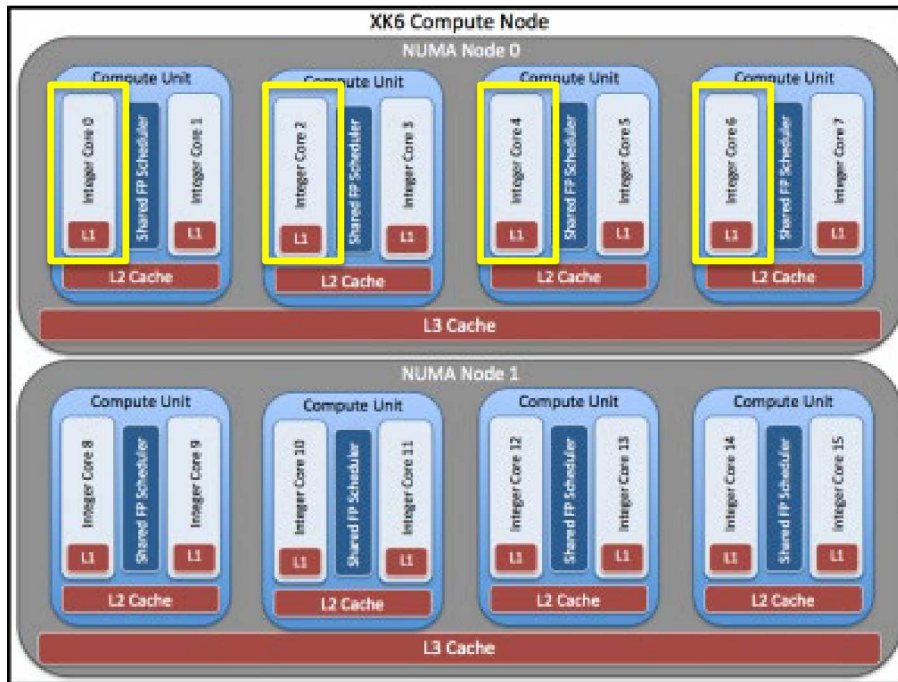
Copper Leaching



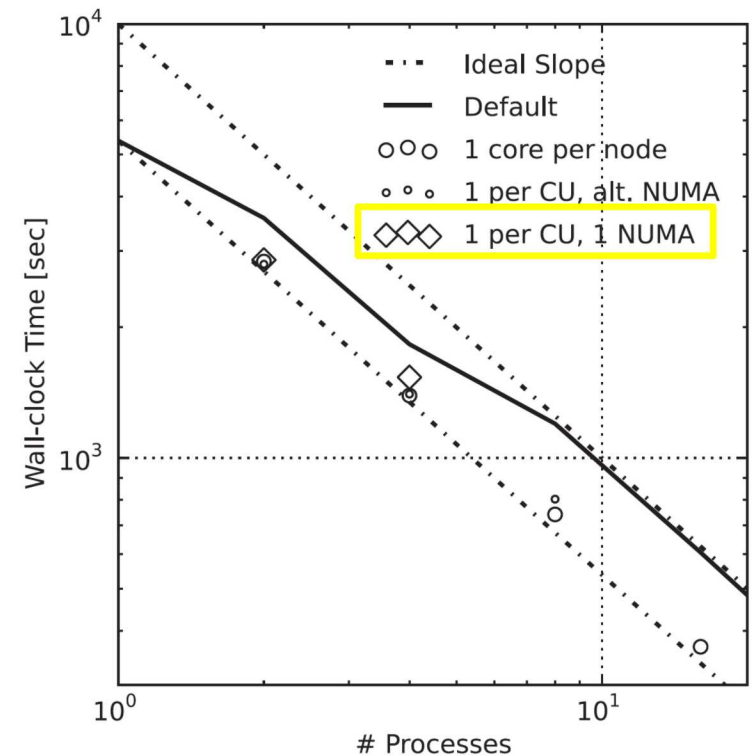
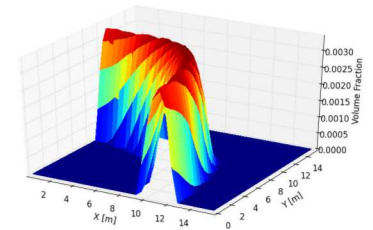
1 to 16 cores on 1 XK6 Node

PFLOTRAN Single Node Performance

XK6 Compute Node with 2 NUMA Nodes



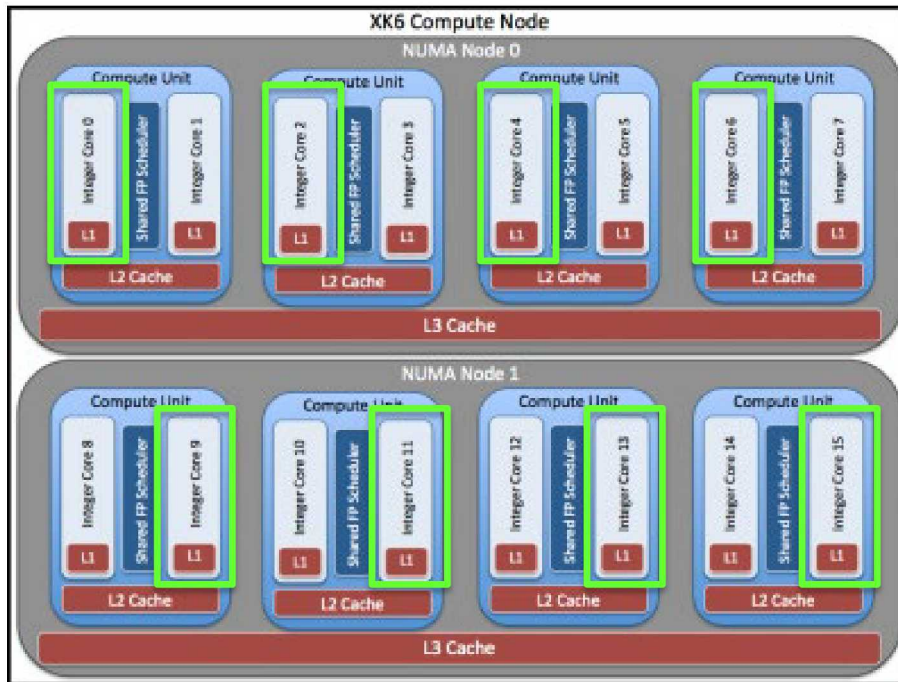
Copper Leaching



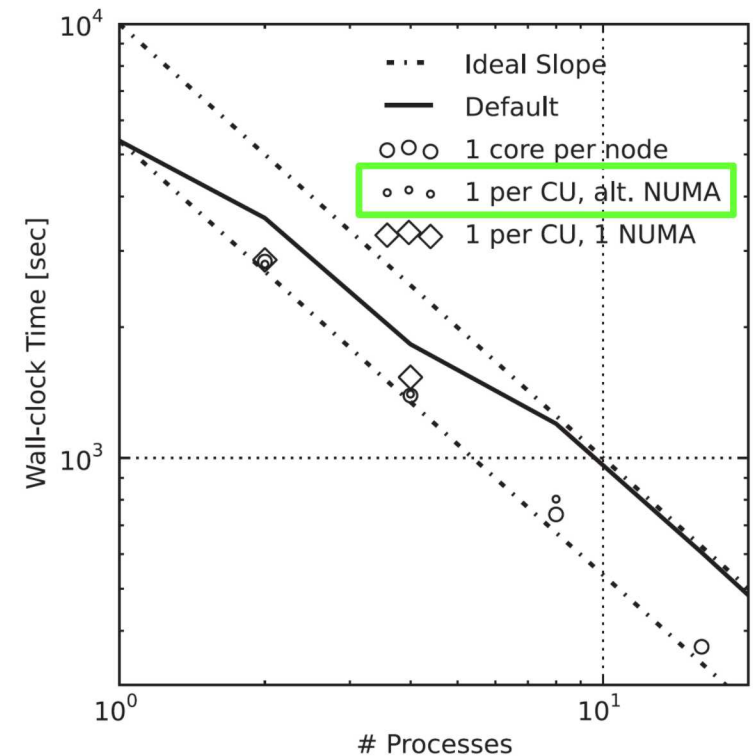
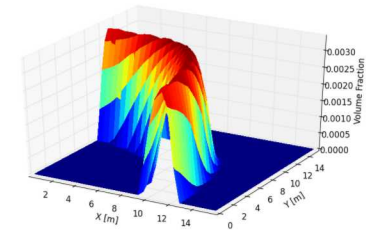
1 to 4 cores on 1 XK6 Node

PFLOTRAN Single Node Performance

XK6 Compute Node with 2 NUMA Nodes



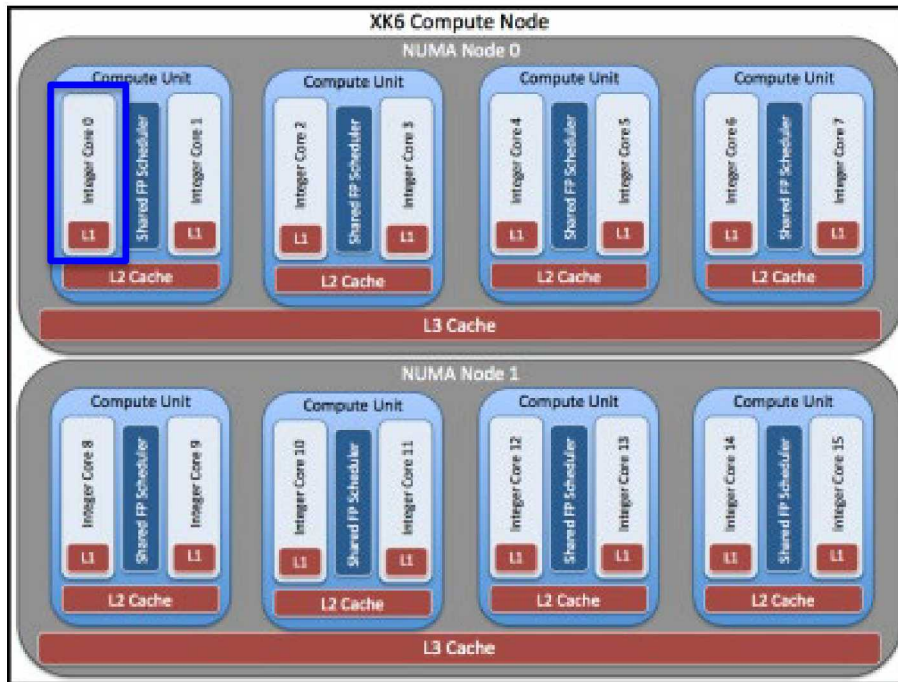
Copper Leaching



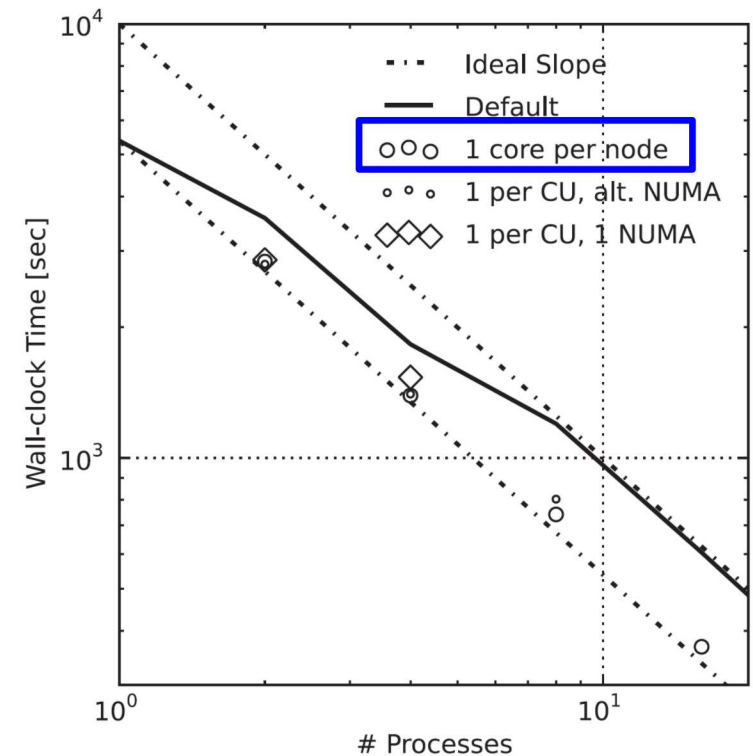
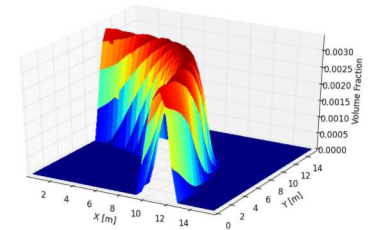
1 to 8 cores on 1 XK6 Node

PFLOTRAN Single Node Performance

XK6 Compute Node with 2 NUMA Nodes



Copper Leaching



1 to 16 cores on 1 to 16 XK6 Nodes

PFLOTRAN Parallel Performance

- Helpful guidelines for good parallel performance
 - $\geq 10,000$ degrees of freedom per process
 - Ratio of computation to communication tends to be too low below 10K
 - Caveat: complex chemistry can violate this metric
 - $\leq 10,000$ processes
 - Linear Krylov solvers do not scale well beyond 10,000 processes
 - $10,000 \times 10,000 = 100,000,000$ dofs max

Questions?