ParFlow solvers and running in parallel

ParFlow Short Course

Module 6

Learning Objectives:

At the end of this module students will understand:

- The basics of ParFlow Newton Krylov approach
- How ParFlow parallelizes the problem
- How to decide how many processors to run on
- How to change solver settings in a ParFlow run
- The most common solver settings to change and why you might want to change them.
- What the Kinsol log file is
- How to use the Kinsol log file to diagnose model performance issues

ParFlow uses a standard suite of solver packages and is very efficient

- Designed to be **parallel** from the ground-up using an object-oriented framework

- Newton Krylov nonlinear approach via the Kinsol solver package (**globally implicit**=*integrated*)

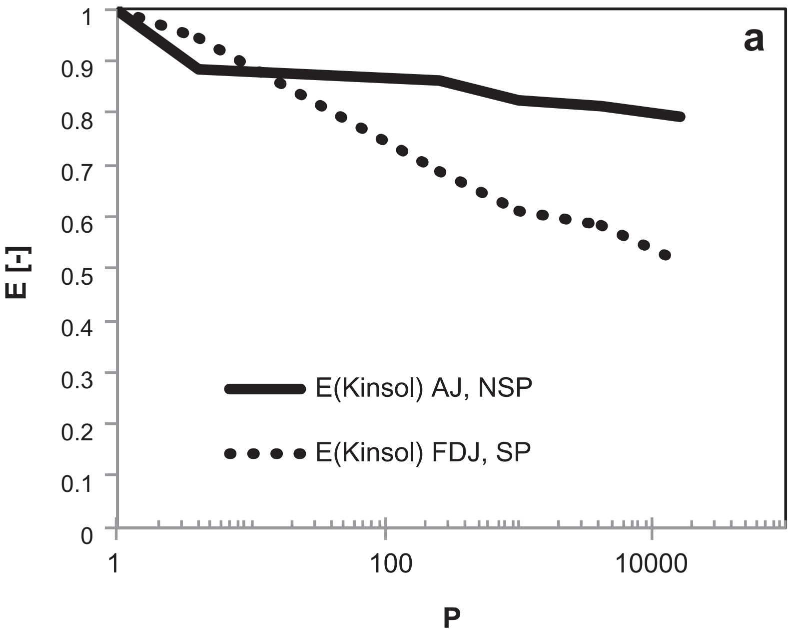
- Multigrid-preconditioned linear solver using the Hypre package

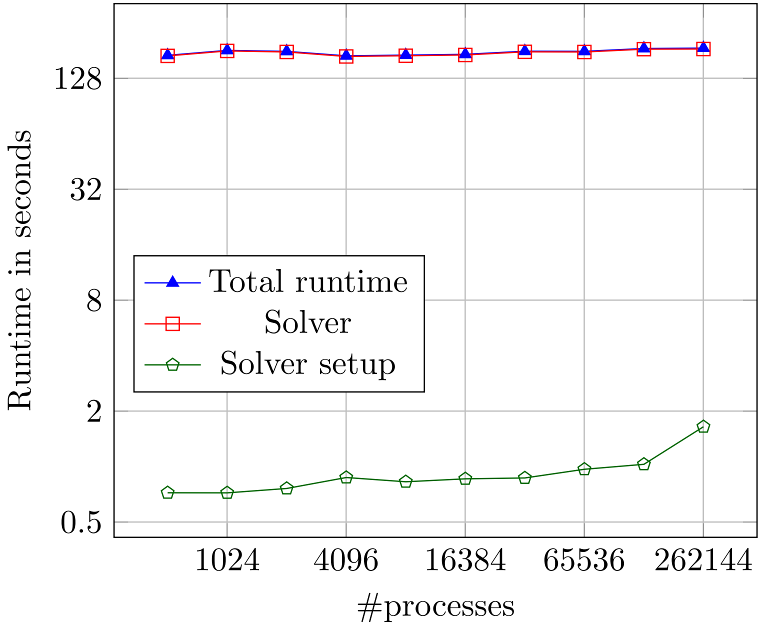
- **Physics-based** preconditioning based on analytical Jacobian to accelerate convergence

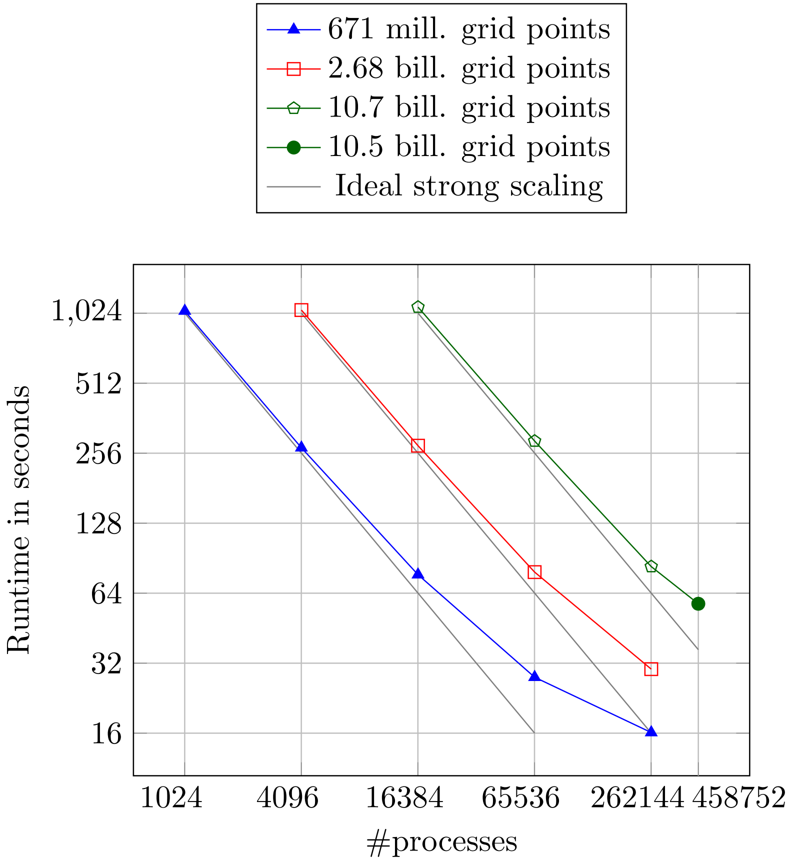
ParFlow demonstrates excellent parallel scalability

Weak scaling, where the problem size increases with the number of processors

Strong scaling, where the problem size remains constant







Maxwell, AWR 2013

Fonseca, arXiv:1702.06898 [cs.MS]

Parallelization: splitting your problem up across multiple processors

- Domain parallelized by specifying number of processor divisions in x,y,z
- Parallelization done on computational domain
- Done using P,Q,R values
  - Total processors=P*Q*R
  - Domain divided by nx/P, ny/Q, nz/R
- Load balancing issues
- Usually keep R as 1 and split in the x and y directions only

Example of Parallelization

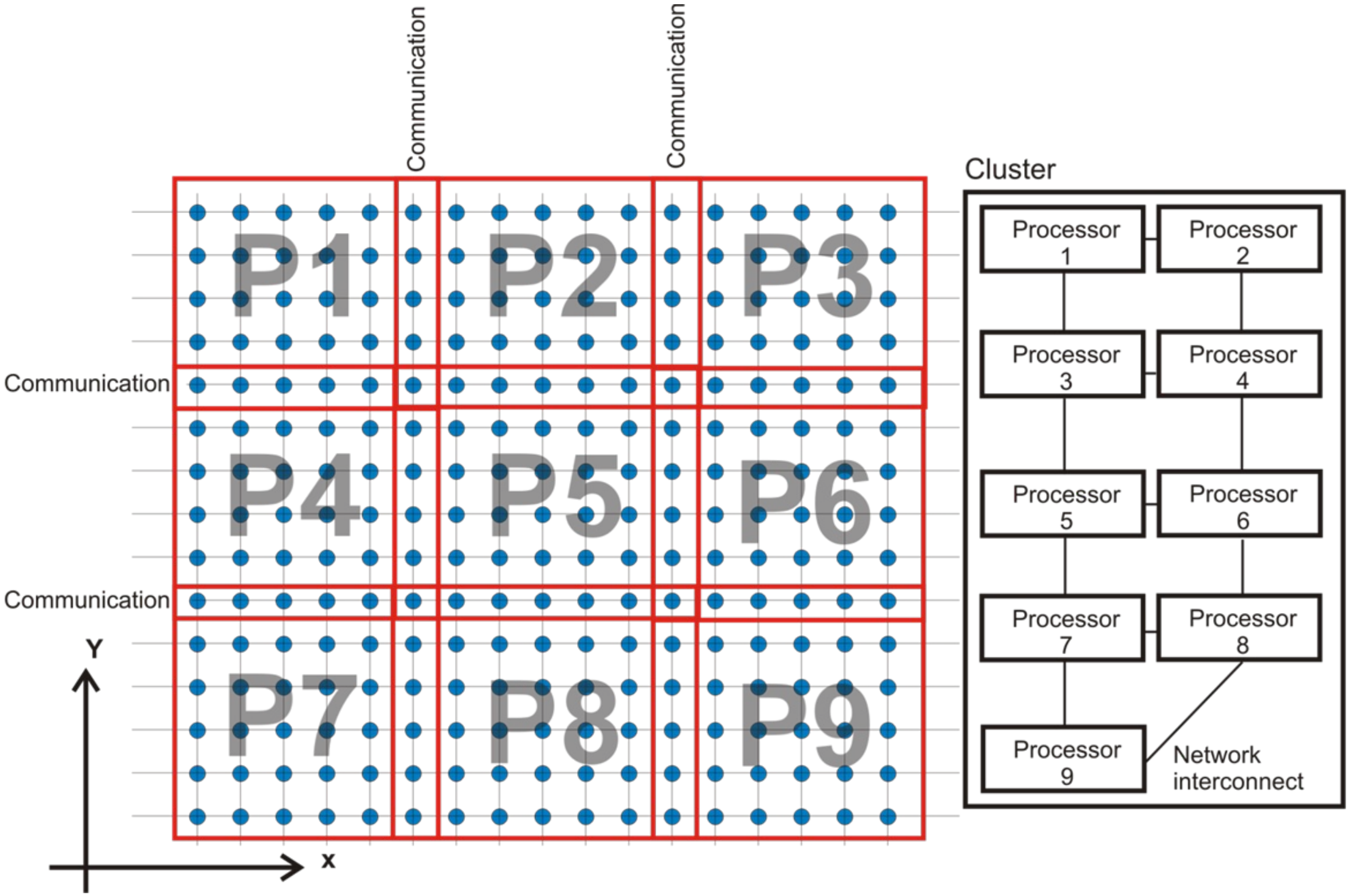
```
model.Process.Topology.P = 1
model.Process.Topology.Q = 1
model.Process.Topology.R = 1
```

Single processor simulation,
P,Q,R are integer values

```
model.Process.Topology.P = 4
model.Process.Topology.Q = 2
model.Process.Topology.R = 1
```

Eight processor simulation,
P*Q*R=4*2*1=8

Domain decomposition



- Lateral transport processes
- Inter-processor communication required
- “Perfect” parallel scaling can not be obtained
- I/O and load balance still constitute bottlenecks

Handling file I/O in parallel

- ParFlow reads and writes parallel files
- One portion of the file per processor (except for sequential/shared memory build)
- ParFlow binary files (.pfb) must be distributed (split up) before being read in
- ParFlow binary files (.pfb) must be undistributed at the end of the simulation
- Two tools to do this, `model.dist()` and `pfundist`, may be run directly in the python input script.

Distributing Files (input file)

```
model.dist("my.input.file.pfb")
```

Distribute an input file
Must have specified processor topology, can happen anywhere in script before model.run() command

```
pfundist default_over

pfundist my.input.file.pfb
```

First line undistributes an entire run

Second line undistributes a particular file

*** You can dist and undist files using separate scripts outside your main model run or you can do it all in one step**

The Kinsol Log File

This is a very important file! It keeps track of the ParFlow's progress as it converges to a solution and can tell you a lot about how hard it is working and when its getting stuck.

NOTE: If you don't have output files yet the model could still be working to solve the first time step. If you don't have a kinsol log then its not doing anything and you have a problem with your model installation or your inputs

http://parflow.blogspot.com/2015/08/kinsol-logs-and-troubleshooting-slow.html

The Kinsol Log File

```
KINSOL starting step for time 1.000000
scsteptol used:       1e-30
fnormtol  used:       1e-06
KINSolInit nni=    0  fnorm=        2859.631546316827  nfe=     1
KINSol nni=    1 fnorm=        2798.943588424358 nfe=     5
        KINSol nni=    2 fnorm=        1264.208176165057 nfe=     8
KINSol nni=    3 fnorm=        1221.629858300943 nfe=    13
KINSol nni=    4 fnorm=        903.7712374454168 nfe=    15
KINSol nni=    5 fnorm=         408.050165635947 nfe=    18
KINSol nni=    6 fnorm=        363.6154118376153 nfe=    26
KINSol nni=    7 fnorm=        4.635468210055366 nfe=    27
KINSol nni=    8 fnorm=      0.06076322632612002 nfe=    28
KINSol nni=    9 fnorm=     5.862763215189762e-05 nfe=    29
KINSol nni=   10 fnorm=     7.663997383858169e-07 nfe=    30
KINSol return value 1
---KINSOL_SUCCESS


-------------------------------------------------
                     Iteration         Total
Nonlin. Its.:            10             10
Lin. Its.:              121            121
Func. Evals.:            30             30
PC Evals.:               10             10
PC Solves:              131            131
Lin. Conv. Fails:         0              0
Beta Cond. Fails:         0              0
Backtracks:               0              0
-------------------------------------------------
```

This tracks the model residual (fnorm) as the model converges.

Once fnorm falls below the tolerance set in your script *Solver.Nonlinear.ResidualTol* the time step is solved.

If…

1. The maximum number of iterations is reached *Solver.Nonlinear.MaxIter*
2. fnorm stops changing between iterations *Solver.Nonlinear.StepTol*

The time step is failed and ParFlow will half the time step and try again

The Kinsol Log File

- It’s normal for the model to require a lot of iterations to converge when its just getting started

- You should see the number of nonlinear iterations (Nonlin. Its.), linear iterations (Lin. Its.) and function evaluations (Func. Evals.) per time step decreasing as the model gets going.