Parallel PEST Using BeoPEST

Willem A. Schreüder

willem@prinmath.com

Princpia Mathematica Inc

University of Colorado at Boulder

PEST Conference

November 3, 2009

Motivation

- Calculating the Jacobian requires *many* runs
  - Runs are independent
  - Order of runs are not important
  - Runs are idempotent
- Embarrassingly parallel problem
  - Naturally parallel
  - Implemented in ***ppest***
- Beowulf clusters are attainable
  - Office PCs are idle 16 hours per day

Original ***ppest*** limitations

- ***ppest*** master process does all the work
  - select parameters
  - write model input files
  - read model output files
- Issues
  - master bogs down reading and writing files
  - must be on a shared file system
  - file system based communications
    - must be synchronous to be correct, implies slow
    - file system performance can be a bottleneck
- ***ppest*** does not scale well to many machines

Features of BeoPEST

- Smart slaves
  - Master just selects parameters
  - Slave reads and writes model files
- Communication via TCP/IP or MPI
  - Direct, reliable communication
  - Slaves can be on an internet
- Fault tolerant
  - Failed slaves does not effect master
- Heterogeneous
  - Mix Windows/MacOS/Unix/Linux
- Scales well to thousands of slaves

Smart Slaves

- Same executable as master with slave flag
- Reads the same ***.pst*** file as the master
- Waits for master to request a run
  - Receive parameters from master (binary)
  - Writes model input file (local disk access)
  - Runs model (local writes only)
  - Reads model output file (local disk access)
  - Sends observations to master (binary)
- Master and slaves sleep while idle (no CPU)

TCP/IP or MPI

- TCP/IP is well suited to ad hoc clusters
  - Reliable transport protocol
  - Slaves added at any time by connecting to master
  - Failed slave detected by dropped connection
  - Works on LAN and over the internet
- MPI is well suited to supercomputers
  - MPI infrastructure to start and manage slaves
  - Can use fast interconnects
  - Currently not fault tolerant, rarely heterogeneous

Running BeoPEST

- Using file system communications (needs .rmf)
  - ppest foo.pst
- Using TCP/IP (master host sherkhan port 4004)
  - Master: ppest foo.pst /H :4004
  - Slaves: ppest foo.pst /H sherkhan:4004
- Using MPI (1024 processes, directory /tmp/xxx)
  - mpirun -np 1024 ppest foo.pst /M /tmp/

Communications

- Values sent as 8 byte binary numbers
  - 8*Npar master to slave
  - 8*Npar + 8*Nobs slave to master
- One message each way per model run
  - Latency measured in milliseconds
  - Slave does blocking read, starts immediately after the last parameter is received
  - Master does non-blocking read, scans active slaves every 5ms for completion

Run Management

- Master schedules jobs to run on idle slaves
  - Newly added slave immediately put to work
  - Job rescheduled if slave fails
- Managing jobs on master uses little CPU
- Idle slaves uses no CPU
- Naive scheduling algorithm
  - Picks fastest idle slave for next job
  - Next job scheduled as soon as slave becomes idle
  - Jobs run in order

File Management

- Each slave slave has its own writable directory
  - Used by slave to create model input files
  - Used to store model output files
  - Does not have to be visible from master
  - Best if on local slave file system
- Read-only global file system
  - Master and slaves can share *.pst/.tpl/.ins* files
  - Common model files not touched by PEST
  - Ensures files are always consistent
  - A bit tricky to get all the file names right

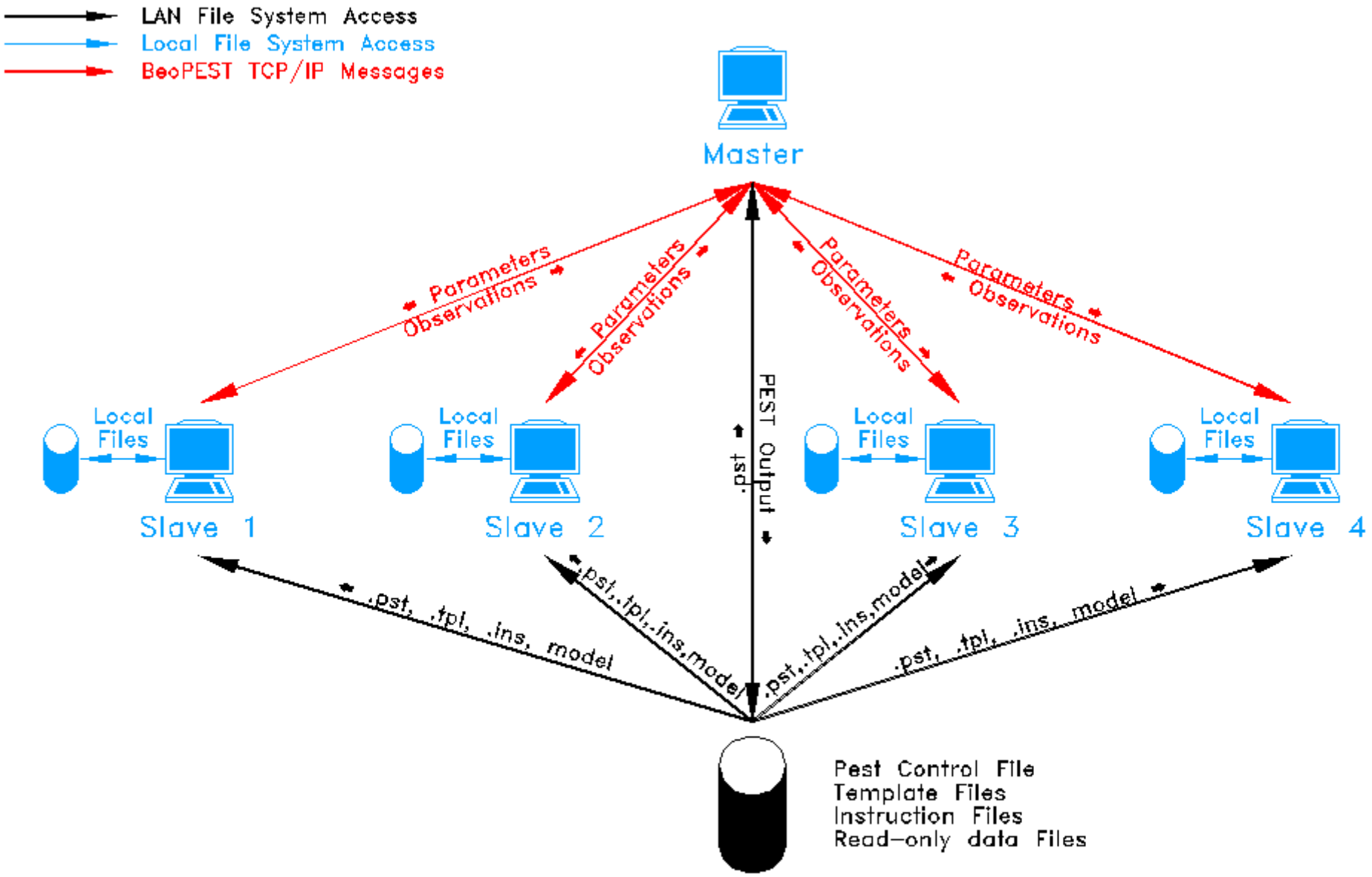

Figure 1. Running BeoPEST on a homogeneous LAN.

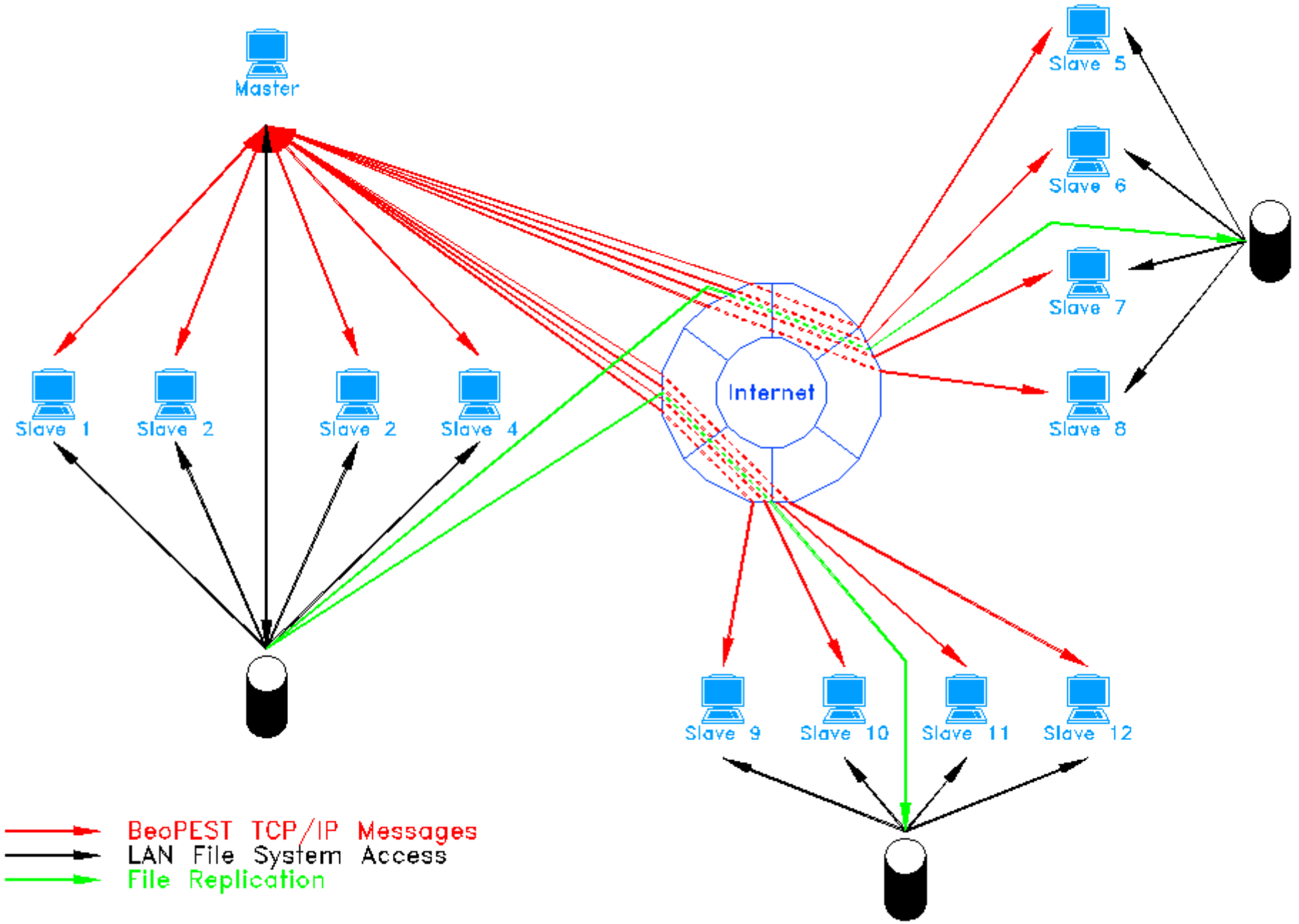

Figure 2. Running BeoPEST across the Internet.

Security

- With BeoPEST/TCP user logs in to each slave
- Programs to run are controlled by ***.pst*** file
- Information send between master and slaves are only parameter and observation values
  - Data are treated as numeric values and cannot be used to inserts symbols
  - Man-in-the-middle modification just corrupts optimization process
- Does require one port to tunnel through firewall

Ongoing work

- Single PEST executable
  - flags control sequential, old parallel, TCP/IP or MPI
- Smarter scheduling and load balancing
  - Very slow slaves can hold up completion
    - schedule jobs on multiple slaves
    - kill others when the first one finishes
  - Look-ahead scheduling (instead of greedy)
- Teaching John how to pronounce *matrix*

Obtaining BeoPEST

- Email: willem@prinmath.com
- Web Site: http://www.prinmath.com/pest/
  - Documentation
  - Source code
  - Executables
    - Linux
    - OSX
- Current version PEST 11.13

And they lived happily ever after.