Madagascar Open-Source Software Package: Interface to Reproducible Research

Sergey Fomel
Jackson School of Geosciences
The University of Texas at Austin

June 30, 2015
Unix Philosophy

“Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface.”

*Doug McIlroy*
Outline

• The tao of Unix

• **Madagascar history and status**

• Unix-style abstraction in Madagascar
  – sfdottest/sfconjgrad
  – sfomp/sfmpi
  – pscons/sfbatch

• Reproducible research
In a Nutshell, Madagascar...

...has had 12,445 commits made by 85 contributors Representing 1,098,245 lines of code
...is mostly written in C with an average number of source code comments
...has a well established, mature codebase maintained by a large development team with stable Y-O-Y commits
...took an estimated 302 years of effort starting with its first commit in May, 2003
Madagascar Contributors
Madagascar Schools
2011年Madagascar计算地球物理暑期学校
2011 Madagascar School on Reproducible Computational Geophysics
Research Pyramid

- Implement
- Test
- Publish
Research Pyramid

Papers

Workflows

Programs
Research Pyramid

2,000 Programs

900 Workflows

8,000 Figures

2,000 Programs

250 Papers

LaTeX

Python

SCons

Unix

C/C++
Outline

• The tao of Unix

• Madagascar history and status

• **Unix-style abstraction in Madagascar**
  – sfdotest/sfconjgrad
  – sfomp/sfmpi
  – pscons/sfbatch

• Reproducible research
sfdottest/sfconjgrad

mathematics  geophysics

• sfconjgrad sfmodeling velocity=vel.rsf \niter=100 x0=m0.rsf < data.rsf > model.rsf

• sfdottest sfmodeling velocity=vel.rsf \mod=model.rsf dat=data.rsf

sfdottest: L[m]*d=1165.87
sfdottest: L'[d]*m=1165.87
sfdotest: fork/exec

random numbers

sfmodeling adj=n

random numbers

sfmodeling adj=y
sfomp/sfmpi

- sfomp sfmodeling split=1 join=2 \ velocity=vel.rsf < shots.rsf > data.rsf
- mpirun –np 100 sfmpi sfmodeling \ split=1 join=2 velocity=vel.rsf \ --input=shots.rsf --output=data.rsf
**sfbatch-pscons**

- **sfbatch**: submit a job to a shared cluster

  ```plaintext```
  sfbatch exe='scons NP=100 data.rsf'
  scons BATCH=1 data.rsf
  ```plaintext```

- **pscons**: parallel scons (wrapper for scons –j)

  ```plaintext```
  Flow('data','shots','modeling',split=[3,'omp'])
  Flow('data','shots','modeling',split=[3,'mpi'])
  Flow('data','shots','modeling',split=[3,1000])
  ```plaintext```
What Does This Command Do?

- sfbatch exe='sfconjgrad mpirun -np 1000 sfmpi sfomp sfmodeling niter=100 vel=velocity.rsf split=3 join=1 --input=dat.rsf --output=mod.rsf'

geophysics
def conjgrad(oper, dat, x0, niter):
    'Conjugate-gradient algorithm for minimizing |A x - dat|^2'
    x = x0
    R = oper(adj=0)[x] - dat
    for iter in range(niter):
        g = oper(adj=1)[R]
        G = oper(adj=0)[g]
        gn = g.dot(g)
        print "iter %d: %g" % (iter+1, gn)
        if 0==iter:
            s = g
            S = G
        else:
            beta = gn/gnp
            s = g+s*beta
            S = G+S*beta
            gnp = gn
            alpha = -gn/S.dot(S)
            x = x+s*alpha
            R = R+S*alpha
    return x
Outline

• The tao of Unix
• Madagascar history and status
• Unix-style abstraction in Madagascar
  – sfdottest/sfconjgrad
  – sfomp/sfmpi
  – pscons/sfbatch
• Reproducible research
“An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.”

(Buckheit and Donoho, 1995)
Reproducible Research

“It is a big chore for one researcher to reproduce the analysis and computational results of another [...] I discovered that this problem has a simple technological solution: illustrations (figures) in a technical document are made by **programs and command scripts that along with required data should be linked to the document itself** [...] This is hardly any extra work for the author, but it makes the document much more valuable to readers who possess the document in electronic form because they are able to track down the computations that lead to the illustrations.” (Claerbout, 1991)
Figure 2: Three different approximations (dashed curves) to the qSV impulse-response surface of Greenhorn Shale (bold curves). On the left is the standard vertical paraxial elliptic approximation. In the center is the horizontal paraxial elliptic approximation. On the right is Muir’s double-elliptic approximation.
Literate Programming
“The basic idea of literate programming is to take a fundamentally different starting point for the presentation of programs to human readers, without any direct effect on the program as seen by the computer. Rather than to present the program in the form in which it will be compiled (or executed), and to intercalate comments to help humans understand what is going on (and which the compiler will kindly ignore), the presentation focuses on explaining to humans the design and construction of the program, while pieces of actual program code are inserted to make the description precise and to tell the computer what it should do.” (van Leeuwen, 1990)
Literate Programming in IPython/Jupyter Notebooks