Programming in Madagascar

Jeff Godwin*

Center for Wave Phenomena
Colorado School of Mines
godwin.jeffrey@gmail.com

Purpose

"Give a man a fish and you feed him for a day. Teach him how to fish and you feed him for a lifetime."
- Someone Wise

Purpose

There are:

- $\approx$ 600 programs in Madagascar
- both seismic and non-seismic
- generic data manipulation tools

Some tasks are not (easily) doable with current tools.

Goals

- Madagascar program design
- Madagascar framework
- Python API
  - SVD
- Demos:
  - SVD
  - MayaVi
- Python and SAGE

Disclaimer

Should you build programs for all of your needs?

Disclaimer

Should you build programs for all of your needs?

NO!

Don't reinvent the wheel

Wheel problems

- Multiply two datasets

Wheel problems

- Multiply two datasets
- Concatenate datasets

Wheel problems

- Multiply two datasets
- Concatenate datasets
- FFT of a dataset

Wheel problems

- Multiply two datasets
- Concatenate datasets
- FFT of a dataset
- Apply a bandpass filter

Wheel problems

- Multiply two datasets
- Concatenate datasets
- FFT of a dataset
- Apply a bandpass filter
- Stolt migrations

Your friend...

sfdoc -k .

Additional resources

- Program examples
- RSFSRC/book/recipes
- User mailing list
- Developer mailing list

Program design

Program architecture

- RSF programs are task-centric

Program architecture

- RSF programs are task-centric
- ONE task per program

Program architecture

- RSF programs are task-centric
- ONE task per program
- Pass data to another program for next task

Program architecture

- RSF programs are task-centric
- ONE task per program
- Pass data to another program for next task
- Data from standard in
- Data to standard out
- Options from command line arguments

Sample problem

Joe wants to apply the newest XYZ filter in the frequency domain, but his RSF data is in the time domain, how should he design his new RSF program?

Possible solutions

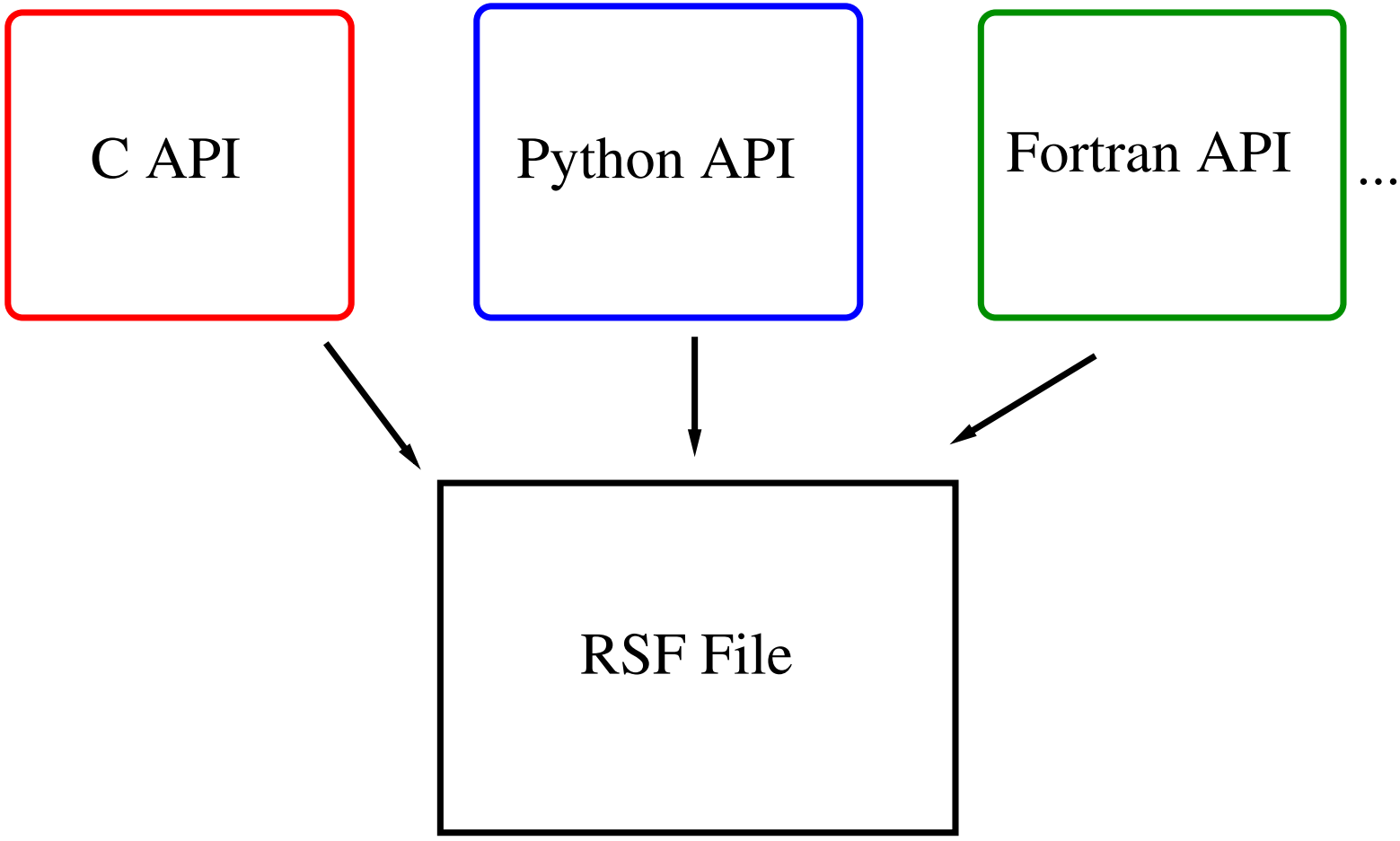
- Write his own code to do the FFT and then apply the filter, and then apply the inverse FFT
- Write his own code to apply the filter to a dataset that has already had the FFT applied, use a C library for the FFT
- Write his code to apply the filter to a dataset that has already had the FFT applied, take the inverse FFT using another program

Possible solutions

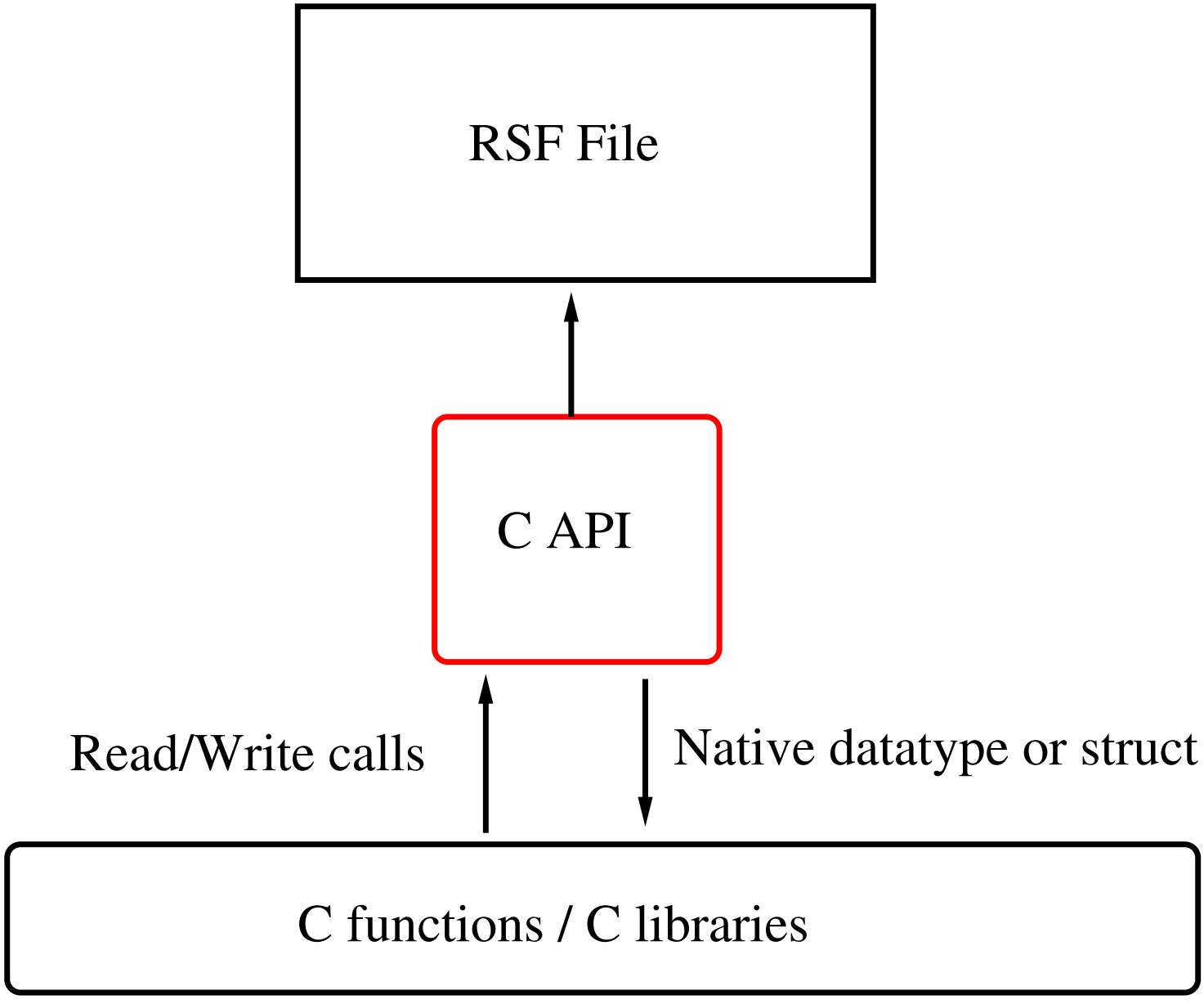
- Write his own code to do the FFT, apply the filter, and apply the inverse FFT
- Write his own code to apply the filter, use a C library for the FFT
- **Write his code to apply the filter to a dataset that has already had the FFT applied, take the inverse FFT using another program**

Madagascar framework

Madagascar framework



API overview



Available API

- C/C++
- **Python**
- Fortran 77
- Fortran 90
- Matlab
- Java
- Octave

API limitations

- Do not fully expose all core C functions

API limitations

- Do not fully expose all core C functions
- Do not expose other RSF programs

API limitations

- Do not fully expose all core C functions
- Do not expose other RSF programs
- Limited communication between APIs

API limitations

- Do not fully expose all core C functions
- Do not expose other RSF programs
- Limited communication between APIs
- Additional dependencies

Python API

Why Python?

- Simple syntax
- Easy to maintain and understand
- Fast function/program prototyping
- Object Oriented (OOP) *Lite*
- Good interface to C/C++
- **Powerful** libraries and packages

80% results with 20% of the effort

When NOT to Use Python

- Performance
- Low-level access
- Significant object overhead

Python RSF program outline

- Documentation (comments)
- Import RSF API
- Initialize RSF command line parser
- Read command line variables
- Declare all input and output RSF files
- Read input data headers
- Read input data sets
- ...
- Create output data headers
- Write output data

Python API Demo: Matrix SVD

Comments rules

- Shebang execution rule : #!/usr/bin/env python
- One line documentation ‴My program does SVD on a 2D matrix ‴
- Block documentation (comments) ”’ line 1 line 2 ... end comments ”’

Comments

```
#!/usr/bin/env python
''' Perform SVD on a matrix using SCIPY.

REQUIRES the PYTHON API, NUMPY AND SCIPY
'''
```

API import

```
# Import RSF API
try:
    import rsf.api as rsf
    import numpy
    import scipy
except Exception, e:
    import sys
    print \
'''ERROR: NEED PYTHON API, NUMPY, SCIPY '''
```

Initialize command line argument parser

```
# Initialize RSF command line parser
par = rsf.Par()
```

Par(ser) rules

- par = Par() # initialize par object
- par.int('n1',1) # get first dimension
- par.float('d1',1.0) # get sampling interval
- par.bool('verb',False) # show verbose output?
- par.string('outname','temp.rsf') # store output where?

Read command line arguments

```
# Read command line variables
vectors = par.bool   ("vectors", False      )
left    = par.string("left"    ,"left.rsf")
right   = par.string("right"   ,"right.rsf")
```

RSF input/output classes

```
input = rsf.Input("in.rsf")
output = rsf.Output("out.rsf")
```

- If no name specified, default to stdin or stdout respectively
- Input.read(numpy.array)
- Output.write(numpy.array)

Declare inputs and outputs

```
# Declare input and outputs
fin = rsf.Input()    # no argument means st
fout = rsf.Output()  # no argument means st
```

Read input headers

```
# Get dimensions of input header or output
n1 = fin.int('n1')
n2 = fin.int('n2')
```

Read datasets

```
data = numpy.zeros((n2,n1),'f') # Note, we

# Read our input data
fin.read(data)
```

Perform SVD

```
# Perform our SVD
u,l,v = numpy.linalg.svd(data)
```

Setup output headers

```
# Perform our SVD
u,l,v = numpy.linalg.svd(data)
```

Write out data

```
# Write output data
fout.write(l)
```

Close open files

```
# Clean up files
fout.close()
fin.close()
```

Python demos

SVD

- Location: book/rsf/programming/samples/svd
- Program: sfsvd
- Execution: scons view
- Demonstrates: the use of simple numpy program wrapped in the RSF API

RSF + Python + MayaVi = Interactive visualization

- Location: book/rsf/programming/samples/cube
- Program : sfthreedcube
- Execution: scons or scons view, user must close pop-up window
- Demonstrates: How to use more advanced libraries within RSF python programs
- Requires: MayaVi and VTK, see (http://code.enthought.com/projects/mayavi/)

Python and SAGE

What is SAGE?

- A collection of *many, many* scientific packages and libraries
- An interactive GUI for developing programs
- A suite for designing, running and sharing programs
- ... much more!

What is SAGE?

- A collection of *many, many* scientific packages and libraries
- An interactive GUI for developing programs
- A suite for designing, running and sharing programs
- ... much more!

Bottom Line: Use SAGE

Python Scripting

rfile = rsf. <program name>(args)[files]

Python Scripting

rfile = rsf. <program name>(args)[files]

- rfile = rsf.spike($n1 = 150, k1 = 25$)[0]
- filt = rsf.bandpass($fhi = 10$)[*rfile*]

RSF Objects

- some operations are defined (add, subtract, multiply)
- can be converted to numpy arrays by slicing: x = rfile[:]
- can be plotted directly:
  - sfwiggle: filt.wiggle().show()
  - sfgrey: filt.grey().show()

SAGE Demo

SAGE Demo

- Location: samples/sage
- Program: None
- Execution: Run SAGE notebook, and upload rsf_demo.sws to worksheet
- Demonstrates: Basic functionality with SAGE
- Requires: SAGE, see (http://sagemath.org)

Conclusions

- Madagascar framework is (relatively) straightforward
- Various APIs provide choice
- Python API is simple, and powerful
- SAGE = lots of possibilities