Centre for Energy Geoscience

THE UNIVERSITY OF WESTERN AUSTRALIA

Leveraging Madagascar for Reproducible Large-scale Cluster and Cloud Computing

Jeffrey Shragge and Toby Potter

The University of Western Australia

Jeffrey.shragge@uwa.edu.au

Shragge and Potter, 2016

Geophysics and Big Data

Industrial-scale Computing

- Revenue drivers:
  - Oil and gas production
- Justify ownership of industrial-scale compute resources
  - 3D TTI RTM, multiparam FWI

Shragge and Potter, 2016

“SME* Computing”

- Revenue drivers:
  - Students, research outputs
- Difficult for SMEs to own large-scale resources
  - Public clusters (competitive)
  - Cloud computing ($$$)
- Goal: open-source, license-free research platform

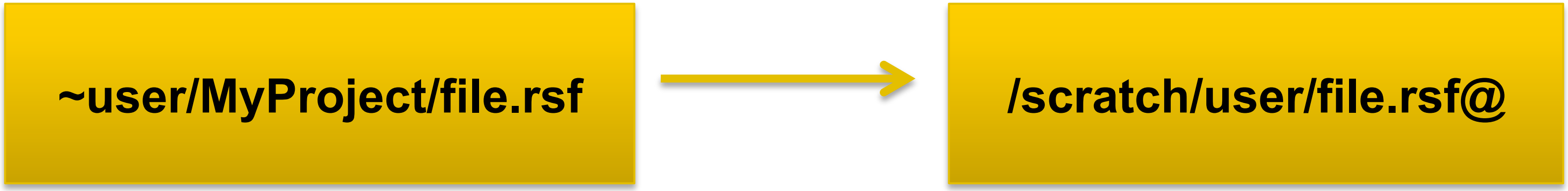
*SME = Small-Medium Enterprise

Outline

- Madagascar: A Quick Refresher
- Madagascar for Cluster-scale Research
- Madagascar in the Cloud

What is Madagascar (M8R)?

- Framework for reproducible computational experiments
- General multi-dimensional data analysis package
- Data files use a regularly sampled format (**RSF**)
  - ASCII metadata file linked to binary data file



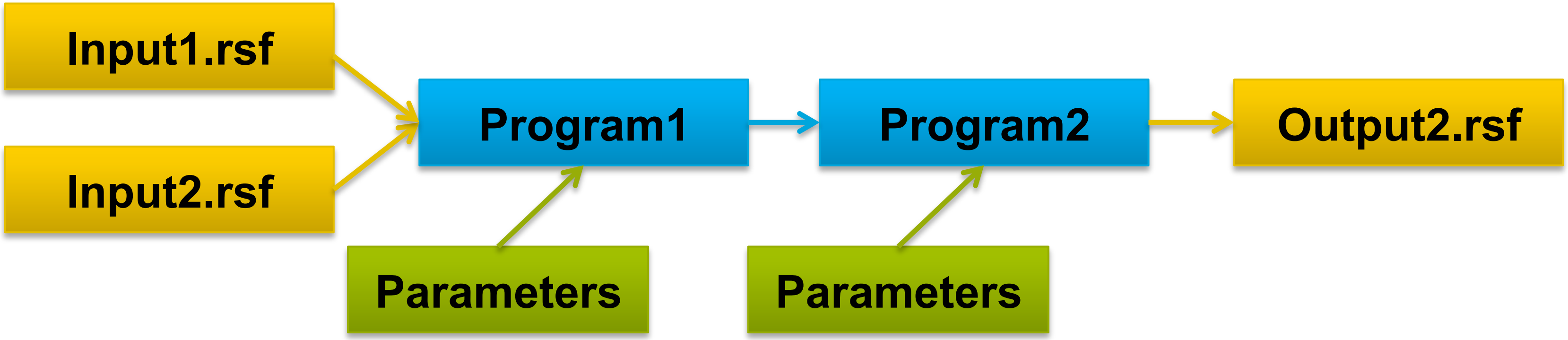
~user/MyProject/file.rsf

/scratch/user/file.rsf@

- M8R programs use common API for file I/O and parameter passing
  - Generic data handling tools for RSF format
  - Specialized / domain-specific tools (e.g., Kirchhoff prestack migration)
  - Growing user-contributed program base (i.e., research outputs)
  - API: C/C++, f90, Python, Java, Matlab/Octave

What is Madagascar (M8R)?

- Data files used in processing flows with I/O linked by common API
  - Interchangeable Flow commands linked by Unix-style pipes: |



- Processing flows written as *SConstruct* scripts
  - Python syntax with Madagascar project extensions

- Use software construction (*scons*) package to run *SConstruct* flows
  - Assess which parts of are up-to-date, and which need to be rerun

Shragge and Potter, 2016

Jeffrey.shragge@uwa.edu.au

M8R *scons* Extensions – *myproj.rsf*

Object	Description
Flow()	Processing flow command linking input/output files, parameters and programs
Plot()	Generate an intermediate plot files
Result()	Generate a final plot file (i.e. for LaTeX manuscript)
Fetch()	Retrieve data file from remote server (ssh)

- Generic *SConstruct* usage

Flow(Target files, Source files, Commands)

- Example of *SConstruct* usage:

```
Flow(output,’input1 input2’,
        ‘sfmath other=${SOURCES[1]} output=“input+other” ’)
```

- Interpreted output

```
< input1.rsf /path/to/sfmath a=${SOURCES[1]} output=“input+a” > output.rsf
```

Shragge and Potter, 2016

Jeffrey.shragge@uwa.edu.au

SConstruct Example – Serial Looping

```
from rsf.proj import *          # . . Import Madagascar project rules

sline = range(0,1000,1)         # . . Set up integer array
# . . Loop over array of 1000 objects

for iss in sline:
    stag = ’04%d’ %iss
    Flow(‘image’+stag,’data’+stag,’my_migration_code par1=… ‘)


# . . Add together object
Flow(‘image’,map(lambda x: ‘image-%04d’,’add ${SOURCES[1:1000]}‘)
End()                           # . . Additional Madagascar framework commands
```

Outline

- Madagascar: A Quick Refresher
- Madagascar for Cluster-scale Research
- Madagascar in the Cloud

M8R *scons* Extensions – *mycluster.py**

Object	Description
Flow()	Processing flow command linking input/output files, parameters and programs
Plot()	Generate an intermediate plot files
Result()	Generate a final plot file (i.e. for LaTeX manuscript)
Fetch()	Retrieve data file from remote server (ssh)
Cluster()	**Provide information on cluster resource requirements** **Queue name, processors per node, walltime (serial)**
Fork()	**Demarcate parallel section; indicate # of nodes, tasks / node, walltime (parallel)**
Iterate()	**Indicate limit of parallel region**
Join()	**End of Fork() section**

***With acknowledgment to Jeff Godwin, Tongning Yang**

SConstruct Example – Serial Looping

```
from rsf.proj import *          # . . Import Madagascar project rules


sline = range(0,1000,1)         # . . Set up integer array
# . . Loop over array of 1000 objects

for iss in sline:
    stag = ’04%d’ %iss
    Flow(‘image’+stag,’data’+stag,’my_migration_code par1=… ‘)


# . . Add together object
Flow(‘image’,map(lambda x: ‘image-%04d’,’add ${SOURCES[1:1000]}‘)
End()                           # . . Additional Madagascar framework commands
```

SConstruct Example – Parallel Looping

```
from rsf.cluster import *   # . . Import Madagascar project rules for your cluster
Cluster(name=’my_queue’,time=60,ppn=24)
sline = range(0,1000,1)     # . . Set up integer array

# . . Loop over array of 1000 objects with 50 jobs on each of 20 nodes
Fork(time=10,ipn=50,nodes=20)
for iss in sline:
    stag = ’04%d’ %iss
    Flow(‘image’+stag,’data’+stag,’my_migration_code par1=… ‘)
    Iterate()
Join()
# . . Add together object
Flow(‘image’,map(lambda x: ‘image-%04d’,’add ${SOURCES[1:1000]}‘)
End()                       # . . Additional Madagascar framework commands
```

SConstruct Example – Parallel Looping

```
from rsf.cluster import *   # . . Import Madagascar project rules for your cluster
Cluster(name=’my_queue’,time=60,ppn=24)
sline = range(0,1000,1)     # . . Set up integer array

# . . Loop over array of 1000 objects with 50 jobs on each of 20 nodes
```

SERIAL

```
Fork(time=10,ipn=50,nodes=20)
for iss in sline:
    stag = ’04%d’ %iss
    Flow(‘image’+stag,’data’+stag,’my_migration_code par1=… ‘)
    Iterate()
Join()
```

PARALLEL

```
# . . Add together object
Flow(‘image’,map(lambda x: ‘image-%04d’,’add ${SOURCES[1:1000]}‘)
End()                       # . . Additional Madagascar framework commands
```

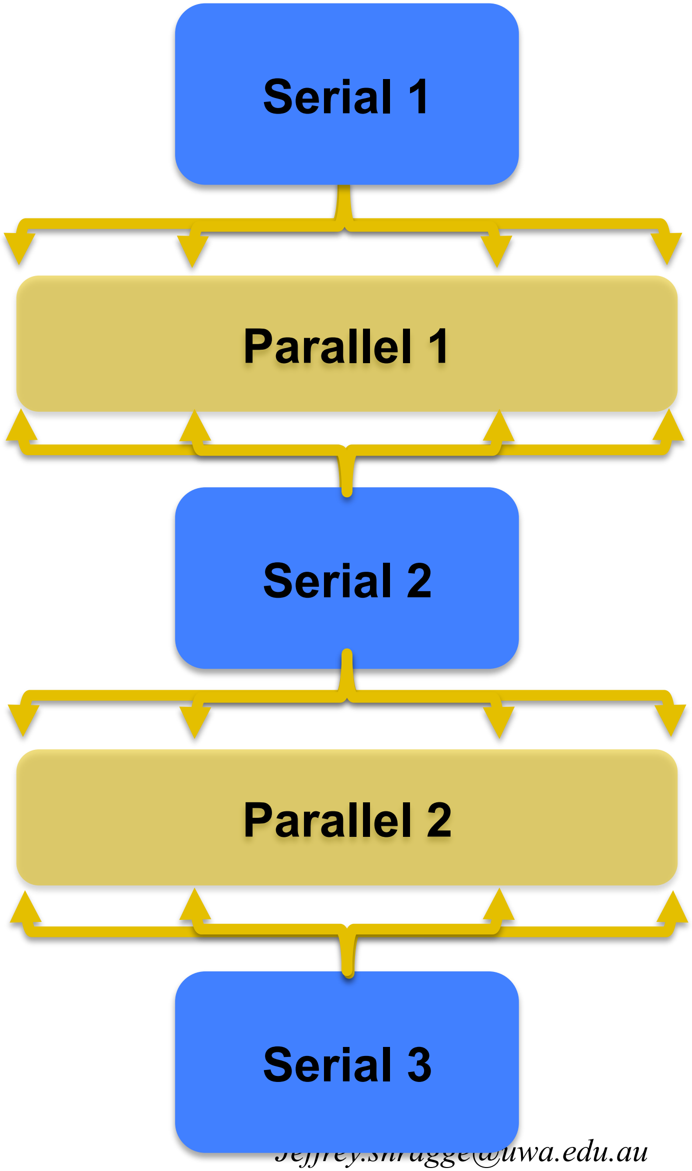
SERIAL

Shragge and Potter, 2016

Jeffrey.shragge@uwa.edu.au

Parallel Execution

- *mycscons* script scans *SConstruct* file to find serial (S) and parallel (P) sections
- Generates submission scripts for each serial and parallel regions
  - Scheduler dependent
- Submits with wait dependencies (-W)
  - Each section starts upon successful completion of previous sections



Serial 1

Parallel 1

Serial 2

Parallel 2

Serial 3

Madagascar and Cluster Parallelism

Design goal	Reason / Example	M8R Scorecard
Allows multicore CPU, GPU, …	Access cluster parallelism	✔
Allow parallelism at scripting level	High degree of data parallelism	✔
Straightforward adaptation for different cluster schedulers	OpenPBS / PBS Pro / SLURM	✔
Scalable	Process 100s of scripted jobs concurrently	✔
Automated Scripting	Optimal use of researcher's time	✔
Failure Handling	Restart at failure point (*scons*)	---
Fault Tolerance	Diagnose failure and adaptively restart	✖

Shragge and Potter, 2016

Outline

- Madagascar: A Quick Refresher
- Madagascar for Cluster-scale Research
- Madagascar in the Cloud

Motivation: 3D Full-wavefield Modelling

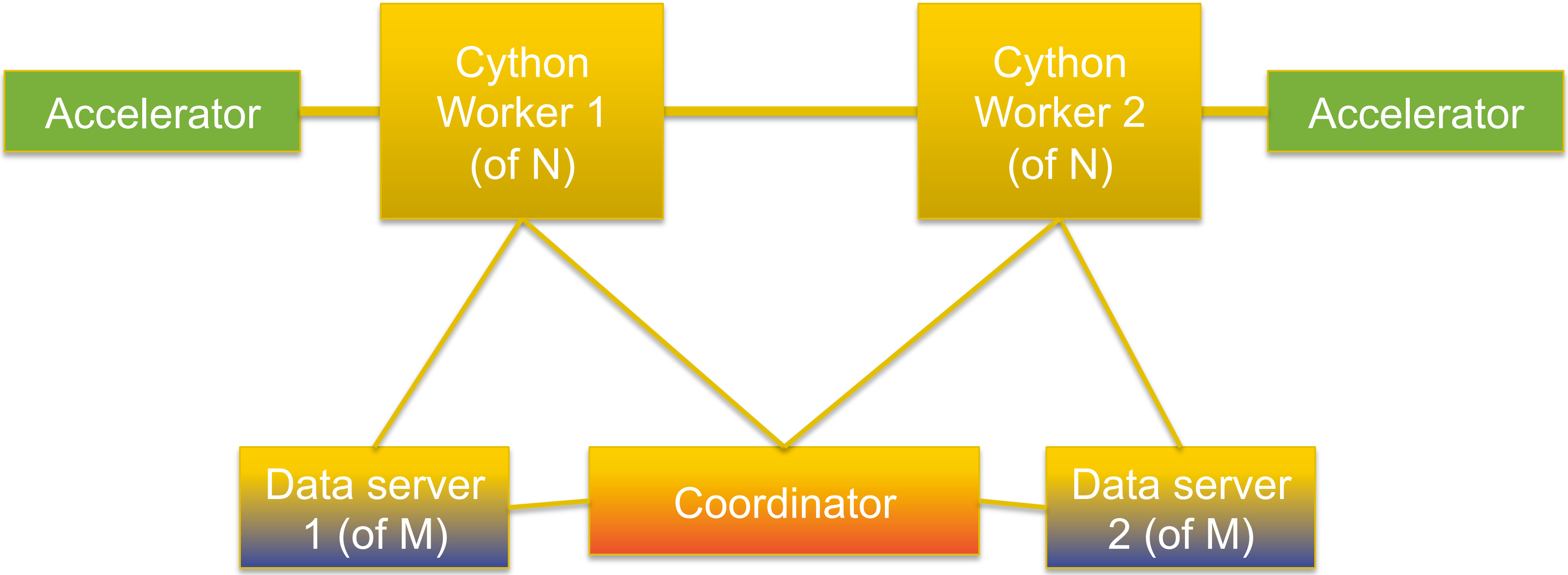
- Multi-year project developing large-scale model of a region of Australia's Northwest Shelf
- Simulate 3D synthetic seismic survey data
  - Requirements: $10^{8\text{-}9}$ core hours
  - Challenging public computing request

- Investigate use of "burst" cloud computing at dynamic pricing
- How can we extend M8R framework to operate on commercial cloud resources with highly variable resource allocations?

Key Design Choices: Toward Fault-tolerance

- Object-oriented actor model with IPython parallel framework
  - Easy-to-deploy under multiple cluster managers/platforms
  - Platform agnostic
- Distributed / streaming I/O (scalability)
  - Data servers with HDF5 container back end
  - Include HDF5 to RSF option
- Message passing (improving fault tolerance)
  - ZeroMQ for fault-tolerant, fast and robust networking
- Data representation (speed)
  - Python Numpy arrays with fast compile-to-C Cython solvers

Design Topology



Accelerator

Cython Worker 1 (of N)

Cython Worker 2 (of N)

Accelerator

Data server 1 (of M)

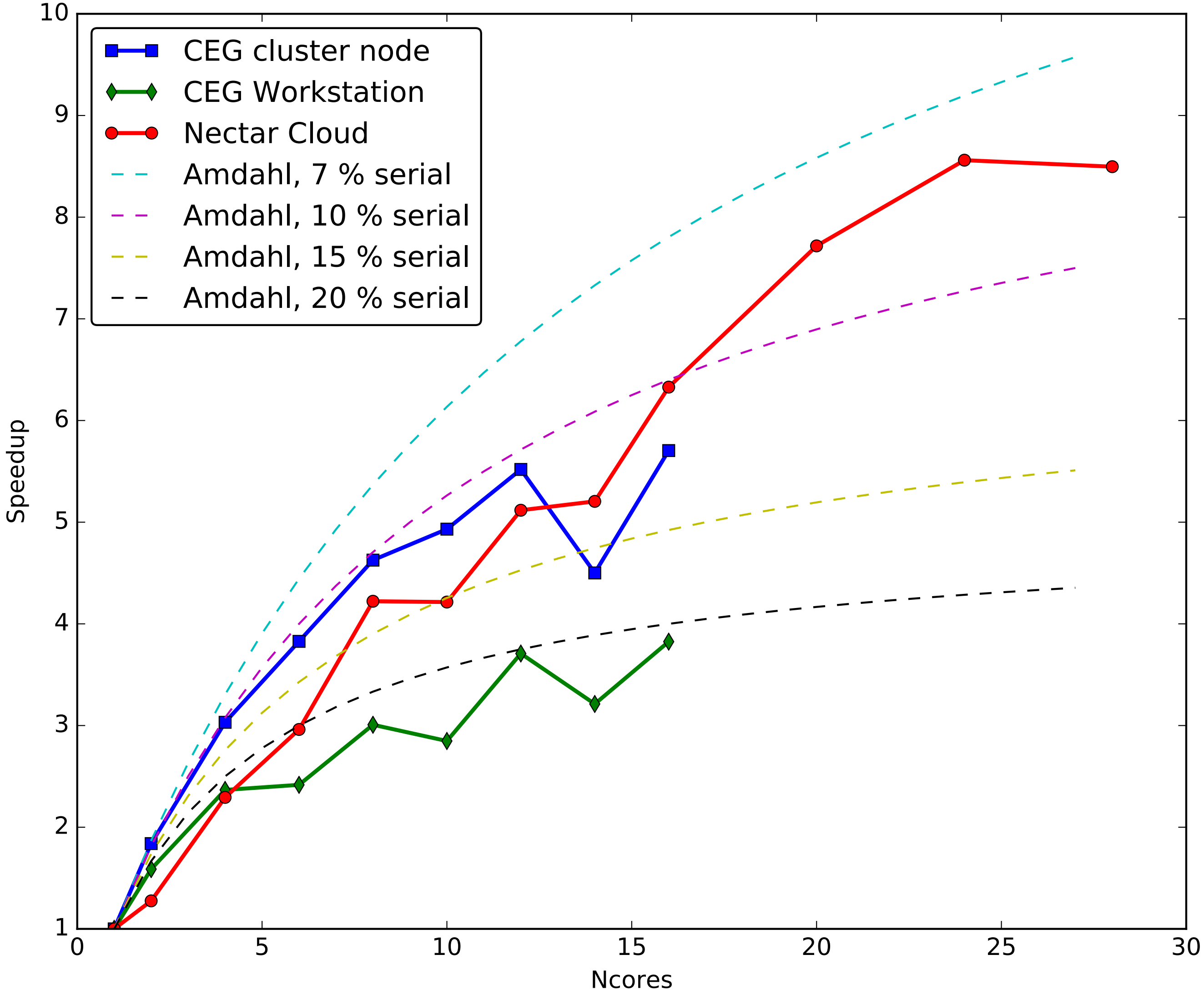
Coordinator

Data server 2 (of M)

Shragge and Potter, 2016

Jeffrey.shragge@uwa.edu.au

Strong Scaling Tests



Shragge and Potter, 2016

Jeffrey.shragge@uwa.edu.au

M8R *scons* Extensions – *mycloud.py* *(in progress)*

Object	Description
Flow()	Processing flow command linking input/output files, parameters and programs
Plot()	Generate an intermediate plot files
Result()	Generate a final plot file (i.e. for LaTeX manuscript)
Fetch()	Retrieve data file from remote server (ssh)
Throw()	**Send data to remote server (ssh)**
Cloud()	**Pass information on cloud resource request: disk image, node configuration, queue name, walltime, …**
Fork()	Demarcate parallel section; indicate # of nodes, tasks / node, walltime (parallel)
Iterate()	Indicate limit of parallel region
Join()	End of Fork() section

Concluding Remarks

- Python scripting easily extends M8R to cluster-scale computing
  - Straightforward mark up with little user overhead

- Goal: Extend M8R framework to easily operate on commercial cloud resources with highly variable resource allocations

- Work in extending M8R in the cloud is ongoing:
  - Platform-independent distributed processing framework using next-generation message passing (ZeroMQ) and HDF5.
  - Tests using research cloud computing environment show promise
  - Goal: to develop M8R *scons* wrappers for cloud environments

Centre for Energy Geoscience

THE UNIVERSITY OF WESTERN AUSTRALIA

Leveraging Madagascar for Reproducible Large-scale Cluster and Cloud Computing

Jeffrey Shragge and Toby Potter

The University of Western Australia

Jeffrey.shragge@uwa.edu.au

Shragge and Potter, 2016