Leveraging Madagascar for Reproducible Large-scale Cluster and Cloud Computing

Toby Potter and Jeffrey Shragge
The University of Western Australia
HPC in Exploration Geophysics

- Full Waveform Inversion
  - Sirgue et al. (2010)
  - Up to PB of data
  - Millions of core-hours
  - Short turnaround

Reverse Time Migration

tgs.com
3D Wavefield propagation

- Finite Difference Time Domain approximations for wavefield modelling
  - Can approximate to high order accuracy
  - Computationally efficient and simple to implement
The future of seismic processing?
Cloud computing options

- Top tier cloud computing providers

  - All offer pricing models with dynamic resource allocation
    - Approximately 70% saving compared to on-demand secured resources

- Need 3D wave modelling codes that can adapt to highly variable resource allocations
  - Cloud generally not as fast as bare metal
  - Scalable applications is key to performance
  - Fault tolerance
An ideal cloud processing framework

- Rapid development and easy modification
  - Researchers don’t want to spend weeks finding memory errors
- Maps to many job engines
  - Generic launcher
- Resistant to node loss
  - Fast storage and retrieval of state
  - Fault tolerant
- Platform Agnostic
  - Not wedded to any one platform
- Adapts to dynamic resource allocation
- Fast
  - Fortran-like performance

http://www.pc-os.org
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 \textit{SConstruct} scripts
  - Declarative Flow specification, you specify dependencies, not order of execution
  - Python syntax with Madagascar project extensions

- Use software construction (\textit{SCons}) package to run \textit{SConstruct} flows
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
## M8R scons Extensions – *mycluster.py*

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

*With acknowledgment to Jeff Godwin, Tongning Yang*
Towards the ideal cloud processing framework

- Object-oriented actor model in a high level language
  - Using Python with *jupyter* framework
  - Julia?

- Maintaining and modifying state (speed)
  - Python Numpy arrays – 64 bit addressing
  - Fast Cython solvers

- Message passing
  - ZeroMQ routers for fault-tolerant, fast and robust networking

- Data servers for distributed IO
  - Flexible IO backends, e.g HDF5, RSF
  - Enables streaming IO from actors
  - Storage and replay of state from file or memory
Isn’t Python slow?

• Numpy arrays: Just contiguous allocations of memory
• Many Numpy operations call optimised C libraries
• Most compute time is spent doing derivatives
• Cython compiles operations to C

\[ \frac{1}{c^2} \frac{\partial^2 u}{\partial t^2} = \nabla^2 u + f \]
Design topology

- 1 Coordinator
- For any collection of machines
  - Any N number of workers
  - Any M number of data servers
Strong scaling with ZeroMQ in the cloud

- Finite-difference time-domain
- Physics: acoustic wave equation
- Grid size: $512^3$
- Platform: Openstack

www.nectar.org.au
## M8R scons Extensions – *mycloud.py* (in progress)

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow()</td>
<td>Processing flow command linking input/output files, parameters and programs</td>
</tr>
<tr>
<td>Plot()</td>
<td>Generate an intermediate plot files</td>
</tr>
<tr>
<td>Result()</td>
<td>Generate a final plot file (i.e. for LaTeX manuscript)</td>
</tr>
<tr>
<td>Fetch()</td>
<td>Retrieve data file from remote server (ssh)</td>
</tr>
<tr>
<td>Throw()</td>
<td><strong>Send data to remote server (ssh)</strong></td>
</tr>
<tr>
<td>Cloud()</td>
<td>Pass information on cloud resource request: disk image, node configuration, queue name, walltime, …</td>
</tr>
<tr>
<td>Fork()</td>
<td>Demarcate parallel section; indicate # of nodes, tasks / node, walltime (parallel)</td>
</tr>
<tr>
<td>Iterate()</td>
<td>Indicate limit of parallel region</td>
</tr>
<tr>
<td>Join()</td>
<td>End of Fork() section</td>
</tr>
</tbody>
</table>
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 engine that leverages ZeroMQ and HDF5.
  – OpenMP-like parallelism observed, with around 5% serial fraction, needs more work
  – Goal: to develop M8R SCons wrappers for cloud environments
UWA Geophysics

Geophysics is the study of the earth, oceans, atmosphere and beyond with the quantitative methods of physics, math and computer science.