PRUNE: A Preserving Run Environment for Reproducible Scientific Computing

-Peter Ivie
• "[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]" – Jon Claerbout
Verify and Extend

• Don’t re-invent the wheel

• Stand on the shoulders of giants
PRUNE features

• Designed for Big Data
• Manage storage and compute resources
• Reproducible workflow specifications
• Share workflow with others
• Reshare changes back
• User defined granularity
Accepted philosophy

- Libraries
- Hardware
- Network
- System Administrators
- Remote Collaborators
- Graduated Students
Proposed philosophy

Preserve Later

Design

Execute

Observe

Share/Publish

Preserve

Preserve First

Design

Preserve

Execute

Share/Publish

Observe

Unpreserve
Differences

- Git: User decides when to preserve
Differences

- Git: User decides when to preserve
- Preserve ALL specification changes
Differences

- Git: User decides when to preserve
- Preserve ALL specification changes
- Git: Code Commits separate from Code Execution
Differences

- **Git:** User decides when to preserve
- **Preserve ALL specification changes**

- **Git:** Code Commits separate from Code Execution
- **System Manages ALL computation**
Differences

- Git: User decides when to preserve
- Preserve ALL specification changes
- Git: Code Commits separate from Code Execution
- System Manages ALL computation
- Remove unneeded items later on
What to Preserve

**Command**: `do < in.txt in.dat > out.txt o2.txt`

- **arguments**: `[file_id1, file_id2]`
- **parameters**: `['in.txt', 'in.dat']`
- **returns**: `['out.txt', 'o2.txt']`
- **results**: `[file_id3, file_id4]`
- **environment**: `envi_id1`
Overview

```
F1 = file_add( filename='./observed.dat' )
export( [ T7[1] ], filename='./plot.jpg' )

T6 = task_add( args=[ T4[0] ], params=['input_data'],
               cmd='analyze < in_data > out_data',
               returns=['out_data'], environment=E2 )

E2 = envi_add( type='EC2', image='hep.stable' )

T7 = task_add( cmd='plot in1 in2 out1 out2',
               args=[ T5[0], T6[0] ],
               params=['in1', 'in2'],
               returns=['out1', 'out2'], environment=E2 )

E1 = envi_add( type='EC2', image='hep.beta' )
```

Workflow Version #2

```
T4 = task_add( cmd='simulate > output',
               returns=['output'], environment=E1)

F1 = file_add( filename='./observed.dat' )

T6 = task_add( args=[ T4[0] ], params=['input_data'],
               cmd='analyze < in_data > out_data',
               returns=['out_data'], environment=E2 )

T5 = task_add( args=[ F1 ], ...
               (remaining arguments the same as above)

T7 = task_add( cmd='plot in1 in2 out1 out2',
               args=[ T5[0], T6[0] ],
               params=['in1', 'in2'],
               returns=['out1', 'out2'], environment=E2 )

export( [ T7[1] ], filename='./plot.jpg' )
```
Sample code: Merge sort

```python
#!/usr/bin/env python
from prune import client
prune = client.Connect() #Use SQLite3

E1 = prune.env_add(type='EC2',
                   image='ami-b06a98d8')
D1, D2 = prune.file_add( 'nouns.txt', 'verbs.txt' )
```
Sample code: Merge sort

```
########## Sort stage ##########
D3, = prune.task_add( returns=['output.txt'],
    env=E1, cmd='sort input.txt > output.txt',
    args=[D1], params=['input.txt'] )
D4, = prune.task_add( returns=['output.txt'],
    env=E1, cmd='sort input.txt > output.txt',
    args=[D2], params=['input.txt'] )

########## Merge stage ##########
D5, = prune.task_add( returns=['merged_out.txt'], env=E1,
    cmd='sort -m input*.txt > merged_out.txt',
    args=[D3,D4], params=['input1.txt','input2.txt'] )
```
Prune Task

Command: 'do < in.txt in.dat > out.txt o2.txt'

returns: ['out.txt', 'o2.txt']

results: [file_id3, file_id4]
Sample code: Merge sort

```
######### Sort stage #########
D3, = prune.task_add( returns=['output.txt'],
    env=E1, cmd=`sort input.txt > output.txt',
    args=[D1], params=['input.txt'] )
D4, = prune.task_add( returns=['output.txt'],
    env=E1, cmd=`sort input.txt > output.txt',
    args=[D2], params=['input.txt'] )

######### Merge stage #########
D5, = prune.task_add(returns=['merged_out.txt'], env=E1,
    cmd=`sort -m input*.txt > merged_out.txt',
    args=[D3,D4], params=['input1.txt','input2.txt'] )
```
Sample code: Merge sort

```
############################ Execute the workflow #################
prune.execute( worker_type='local', cores=8 )
#prune.execute( worker_type='wq', name='myapp' )

############################ Export #################
prune.export( D5, `merged.txt' ) # Final data
prune.export( D5, `wf.prune', lineage=2 )
```
Sample code: Merge sort

```
###### Execute the workflow #######
prune.execute( worker_type='local', cores=8 )
#prune.execute( worker_type='wq', name='myapp' )

###### Export ######
prune.export( D5, `merged.txt' ) # Final data
prune.export( D5, `wf.prune', lineage=2 )
```
Sharable workflow description file

{"body": {"args": ["f908ff689b9e57f0055875d927d191ccd2d6deef:0", "319418e43783a78e3cb7e219f9a1211cba4b3b31:0"], "cmd": "sort -m input*.txt > merged_output.txt", "env": "da39a3ee5e6b4b0d3255bfe95601890af8079", "env_vars": {}, "params": ["input1.txt", "input2.txt"], "precise": true, "returns": ["merged_output.txt"], "types": []}, "cbid": "e82855394e9dcdee03ed8a25c96c79245fd0481a", "size": 322, "type": "call", "wfid": "a0230143-9b3a-4766-809d-5b7172e9b967", "when": "1476886144.7171359"}

{"body": {"args": ["29ae0a576ab660cb17fb9b14729c7b464fa98cca"], "cmd": "sort input.txt > output.txt", "env": "da39a3ee5e6b4b0d3255bfe95601890af8079", "env_vars": {}, "params": ["input.txt"], "precise": true, "returns": ["output.txt"], "types": []}, "cbid": "f908ff689b9e57f0055875d927d191ccd2d6deef", "size": 241, "type": "call", "wfid": "a0230143-9b3a-4766-809d-5b7172e9b967", "when": "1476886144.7171359"}

{"body": {"args": ["48044131b31906e6c917d857dd1539278c455cf"], "cmd": "sort input.txt > output.txt", "env": "da39a3ee5e6b4b0d3255bfe95601890af8079", "env_vars": {}, "params": ["input.txt"], "precise": true, "returns": ["output.txt"], "types": []}, "cbid": "319418e43783a78e3cb7e219f9a1211cba4b3b31", "size": 241, "type": "call", "wfid": "a0230143-9b3a-4766-809d-5b7172e9b967", "when": "1476886144.7171359"}

{"cbid": "29ae0a576ab660cb17fb9b14729c7b464fa98cca", "size": 144, "type": "file", "wfid": "a0230143-9b3a-4766-809d-5b7172e9b967", "when": "1476886144.7171359"}

```
\text{time} \quad \text{person} \quad \text{year} \quad \text{Way} \\
...```

"
Workflow evolution (US Censuses)

Stage 1
Uncompress (year+fragment)

Stage 2
Normalize (year+fragment)

Stage 3
Split by key (year+fragment+key)

Stage 4
Join fragments (year+key)

Stage 5
Pair by year (year1+year2+key)

Stage 6
Group matches (year1+year2+key)

Stage 7
Filter 1-1 matches (year1+year2+key)
Redefine filter criteria

Stage 1
Uncompress (year+fragment)

Stage 2
Normalize (year+fragment)

Stage 3
Split by key (year+fragment+key)

Stage 4
Join fragments (year+key)

Stage 5
Pair by year (year1+year2+key)

Stage 6
Group matches (year1+year2+key)

Stage 7
Filter 1-1 matches (year1+year2+key)
Redefine match criteria

Stage 1
Uncompress (year+fragment)

Stage 2
Normalize (year+fragment)

Stage 3
Split by key (year+fragment+key)

Stage 4
Join fragments (year+key)

Stage 5
Pair by year (year1+year2+key)

Stage 6
Group matches (year1+year2+key)

Stage 7
Filter 1-1 matches (year1+year2+key)
New key function chosen

Stage 1
Uncompress (year+fragment)

Stage 2
Normalize (year+fragment)

Stage 3
Split by key (year+fragment+key)

Stage 4
Join fragments (year+key)

Stage 5
Pair by year (year1+year2+key)

Stage 6
Group matches (year1+year2+key)

Stage 7
Filter 1-1 matches (year1+year2+key)
Stage 1
Uncompress (year+fragment)

Stage 2
Normalize (year+fragment)

Stage 3
Split by key (year+fragment+key)

Stage 4
Join fragments (year+key)

Stage 5
Pair by year (year1+year2+key)

Stage 6
Group matches (year1+year2+key)

Stage 7
Filter 1-1 matches (year1+year2+key)
New input data

Stage 1
Uncompress (year+fragment)

Stage 2
Normalize (year+fragment)

Stage 3
Split by key (year+fragment+key)

Stage 4
Join fragments (year+key)

Stage 5
Pair by year (year1+year2+key)

Stage 6
Group matches (year1+year2+key)

Stage 7
Filter 1-1 matches (year1+year2+key)
Derivation History = Cachable Results

- Generated File content
  - Run workflow twice
  - Run twice w/memoization
  - Run workflow once

- Meta-data stored
  - Run workflow twice
  - Run twice w/memoization
  - Run workflow once

Stage number completed when size was measured
Execution time cut in half for run #2
Quotas

Disk Space

- Virtual size
- Quota
- Actual size

Time (hours)

Zoom
Scalability

- ~12,000 parallel cores
- ~3 million tasks

- Overhead
  ~1% above native wall clock
Sharing workflow between users...

- time
- person
- year

---

Graphs showing disk space over time, file content, and metadata storage.
http://ccl.cse.nd.edu/research/papers/

- Sample workflows
- http://ccl.cse.nd.edu/software/prune/prune.html
  - Merge sort
  - Pairwise comparisons (US Censuses)
  - High-energy Physics

For more information: pivie@nd.edu

Thank You!