Weaving Abstractions into Workflows

Peter Bui
University of Notre Dame

Programming Distributed Applications

Distributed computing is hard

- Resource management, task scheduling, programming interface

Abstractions

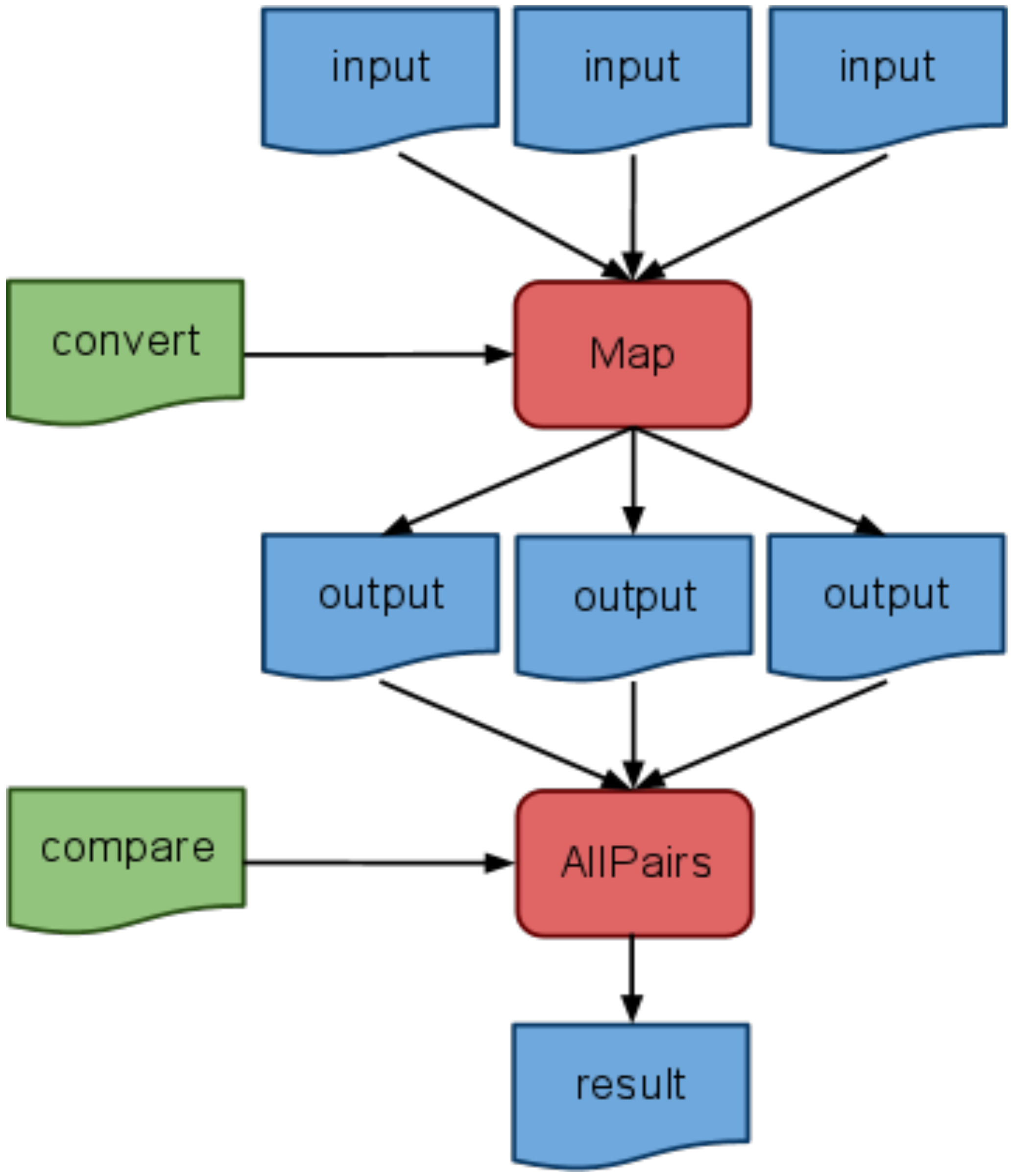
- Structured way of combining small executables into parallel graphs that can be scaled up to large sizes
  - Map-Reduce
  - All-Pairs

What if we want to use multiple abstractions?
What if the abstraction is not available?

Workflow DAGs

Workflows

- Organize execution in terms of directed acyclic graph
  - Makeflow
  - DAGMan
- Pipeline abstractions using graph
- Implement abstractions as nodes and links in a graph



Biometrics Experiment (Makeflow)

```
234437.bit:    /bxgrid/fileid/234437 convert_iris_to_template
        ./convert_iris_to_template /bxgrid/fileid/234437 234437.bit

234438.bit:    /bxgrid/fileid/234438 convert_iris_to_template
        ./convert_iris_to_template /bxgrid/fileid/234438 234438.bit

234439.bit:    /bxgrid/fileid/234439 convert_iris_to_template
        ./convert_iris_to_template /bxgrid/fileid/234439 234439.bit

ap_00_00.txt:  234437.bit 234437.bit compare_iris_templates
        ./compare_iris_templates 234437.bit 234437.bit

ap_00_01.txt:  234437.bit 234438.bit compare_iris_templates
        ./compare_iris_templates 234437.bit 234438.bit

ap_00_02.txt:  234437.bit 234439.bit compare_iris_templates
        ./compare_iris_templates 234437.bit 234439.bit

ap_01_00.txt:  234438.bit 234437.bit compare_iris_templates
        ./compare_iris_templates 234438.bit 234437.bit

ap_01_01.txt:  234438.bit 234438.bit compare_iris_templates
        ./compare_iris_templates 234438.bit 234438.bit

ap_01_02.txt:  234438.bit 234439.bit compare_iris_templates
        ./compare_iris_templates 234438.bit 234439.bit

...
```

Biometrics Experiment (Makeflow)

```
234437.bit:    /bxgrid/fileid/234437 convert_iris_to_template
        ./convert_iris_to_template /bxgrid/fileid/234437 234437.bit

234438.bit:    /bxgrid/fileid/234438 convert_iris_to_template
        ./convert_iris_to_template /bxgrid/fileid/234438 234438.bit

234439.bit:    /bxgrid/fileid/234439 convert_iris_to_template
        ./convert_iris_to_template /bxgrid/fileid/234439 234439.bit

ap_00_00.txt:  234437.bit 234437.bit compare_iris_templates
        ./compare_iris_templates 234437.bit 234437.bit

ap_00_01.txt:  234437.bit 234438.bit compare_iris_templates
        ./compare_iris_templates 234437.bit 234438.bit

ap_00_02.txt:  234437.bit 234439.bit compare_iris_templates
        ./compare_iris_templates 234437.bit 234439.bit

ap_01_00.txt:  234438.bit 234437.bit compare_iris_templates
        ./compare_iris_templates 234438.bit 234437.bit

ap_01_01.txt:  234438.bit 234438.bit compare_iris_templates
        ./compare_iris_templates 234438.bit 234438.bit

ap_01_02.txt:  234438.bit 234439.bit compare_iris_templates
        ./compare_iris_templates 234438.bit 234439.bit

...
```

Manually constructing DAGS is a tedious, error-prone process.

Sophisticated workflows require LARGE DAGs.

DAGs are too low-level.

DAGs are the assembly language of distributed computing.

Biometrics Experiment (Weaver)

```
db   = SQLDataSet('db', 'biometrics', 'irises')
nefs = Query(db, db.c.state == 'Enrolled',
                 Or(db.c.color == 'Blue',
                    db.c.color == 'Green'))

convert = SimpleFunction('convert_iris_to_template')
compare = SimpleFunction('compare_iris_templates')

bits = Map(convert, nefs)

AllPairs(compare, bits, bits, output = 'matrix.txt',
         use_native = True)
```

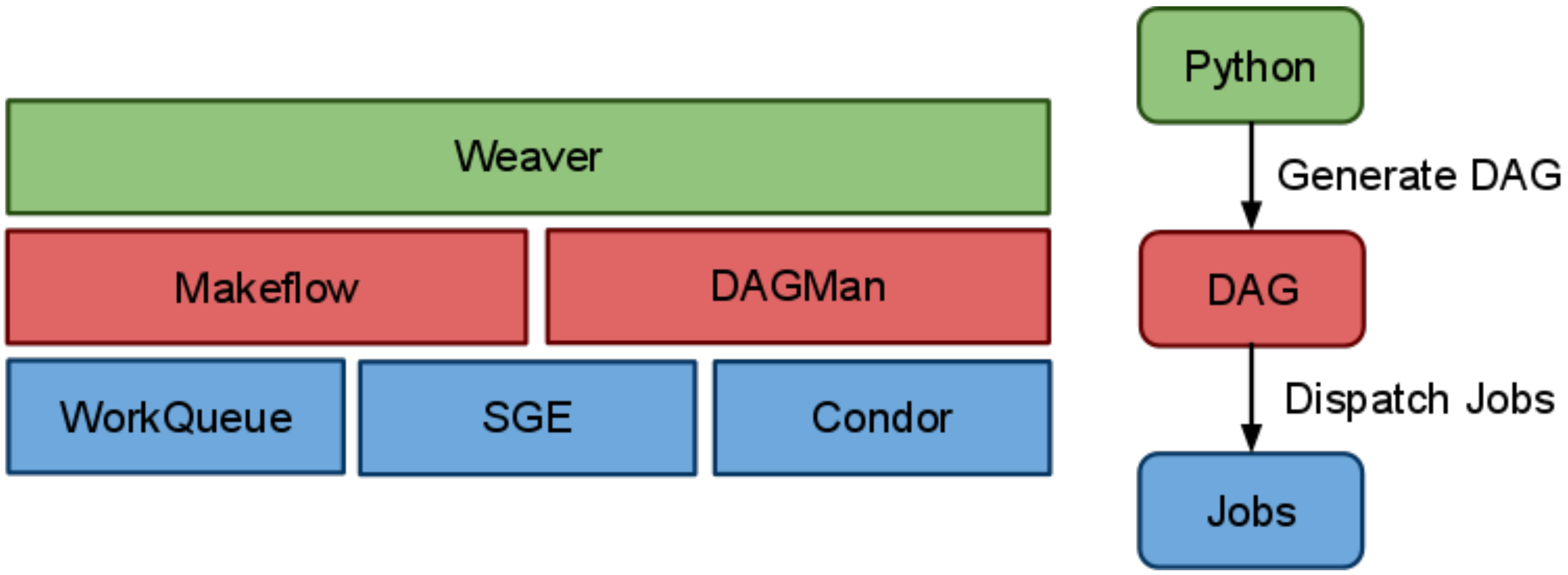
Weaver

High-level framework that allows users to integrate abstractions into their workflows.

Unique Features

- Built on top of *Python* programming language.
- Compiles workflows for multiple workflow managers (*Makeflow*, *DAGMan*).
- Construct generic versions of abstractions as DAGs.

Software Stack



Programming Model

Datasets

- Any *iterable* collection of Python objects.
- Simple ORM interface through Query function.

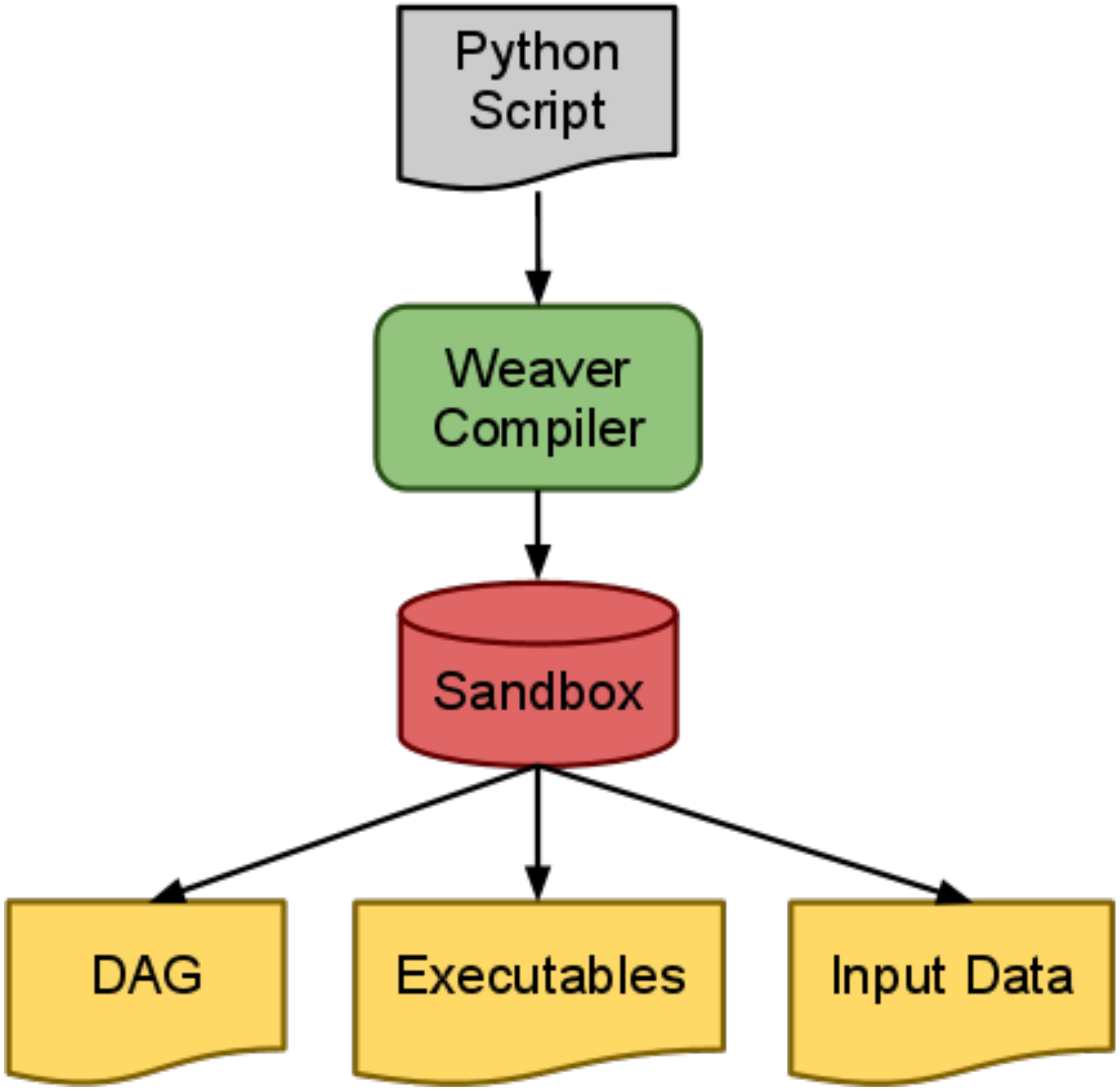
Functions

- Binary executables or Python Functions.

Abstractions

- Describe how functions are applied to dataset.
- Includes: *Map*, *MapReduce, All-Pairs*, *Wavefront*

Compiler


Python Script
Weaver Compiler
Sandbox
DAG
Executables
Input Data

Map-Reduce

```
def wc_mapper(key, value):
    for w in value.split():
        print '%s\t%d' % (w, 1)

def wc_reducer(key, values):
    print '%s\t%d' % (key, sum(map(int, values))

MapReduce( mapper  = wc_mapper,
           reducer = wc_reducer,
           input   = Glob('weaver/*.py'),
           output  = 'wc.txt')
```

Molecular Dynamics Analysis

```
ANALYSIS_ROOT = '/afs/crc.nd.edu/user/h/hfeng/Public/DHFRTS_project/Analysis/'
residue_file  = os.path.join(ANALYSIS_ROOT, 'correlationCode', 'residueList')
residue_list  = [s.strip() for s in open(residue_file)]
residue_tars  = []

def make_residue_archiver(r):
    archiver = Function('tar_residue.py')
    archiver.output_string  = lambda i:     '_'.join(i.split()) + '.tar.gz'
    archiver.command_string = lambda i, o:  './tar_residue.py ' + r
    return archiver

for r in residue_list:
    f = make_residue_archiver(r)
    t = Run(f, '', output = f.output_string(r))
    residue_tars.append(t[0])

comparer = SimpleFunction('dihedral_mutent.sh', out_suffix = 'tar')
comparer.add_functions(Glob(os.path.join(ANALYSIS_ROOT, 'correlationCode',
'*py')))
merger   = SimpleFunction('tar_merge.sh', out_suffix = 'tar')
AllPairs(comparer, residue_tars, residue_tars, output = 'results.tar',
merge_func = merger)
```

Conclusion

Weaver is...

- Workflow compiler and framework.
    - DAGs = Assembly Language
    - Abstractions = SIMD instructions
- Python library for distributed computing.
- Prototyping tool for abstraction builders.

Status

- Work in progress, but usable and working state.
    - Used for biometrics experiments.
    - Foundation for molecular dynamics analysis framework.
- Source code: http://bitbucket.org/pbui/weaver