Phylanx in Python:

The Interactive Distributed Computing Tool of the Future

In [1]:
```
import phylanx
from phylanx.ast import *
import numpy as np

et = phylanx.execution_tree
cs = phylanx.compiler_state()
```

In [2]:
```
# Step 1: Write PhySL code in PhySL
# but execute it with Python
result = et.eval("""
block(
    define(fib,n,
    if(n<2,n,
        fib(n-1)+fib(n-2))
    ),
    fib
    )""",cs,et.var(10))
print(result)
```

```
55.0
```

In [3]:
```
# Step 2: Write Python code that
# gets magically turned into PhySL code.

@Phylanx(debug=True) # biuld with phylanx!
def foo(n):
    print("n=",n)
```

```
define(
    foo,
    n,
    (
        cout("n=", n)
    )
)
```

In [4]:
```
foo(5)
```

```
n= 5
```

Out[4]:
```
<nil>
```

In [5]:
```
arr = np.array([1,2,3,4])
@Phylanx(debug=True)
def dotme(arr):
    return dot(arr,arr)
print(dotme(arr))
```

```
define(
    dotme,
    arr,
    dot(arr, arr)
)
30.0
```

In [6]:
```
import numpy as np
f = np.linspace(0,10,11) # create a simple numpy array
```

In [7]:
```
@Phylanx()
def pr(a):
    print(a) # This prints to the console, so we don't see it.
    return a[1:5] # Slicing works
pr(f)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Out[7]:
```
array([ 1.,  2.,  3.,  4.])
```

In [8]:
```
@Phylanx()
def fib(n):
    if n < 2:
        return n
    f1 = fib(n-1)
    f2 = fib(n-2)
    return f1+f2 # Doesn't work, the underlying PhySL only supports a single return
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
<ipython-input-8-115beb1bb6a9> in <module>()
----> 1 @Phylanx()
      2 def fib(n):
      3     if n < 2:
      4         return n
      5     f1 = fib(n-1)

~/.local/lib/python3.5/site-packages/phylanx-0.0.1-py3.5-linux-x86_64.egg/phylan
x/ast/transformation.py in __init__(self, f)
    592             ast.increment_lineno(tree, actual_lineno)
    593             assert len(tree.body) == 1
--> 594             self.transformation = self.targets[target](tree, kwargs)
    595             self.__src__ = self.transformation.__src__
    596             if self.debug:

~/.local/lib/python3.5/site-packages/phylanx-0.0.1-py3.5-linux-x86_64.egg/phylan
x/ast/transformation.py in __init__(self, tree, kwargs)
     57             for arg in tree.body[0].args.args:
     58                 self.defs[arg.arg] = 1
---> 59         self.__src__ = self.recompile(tree)
     60
     61     def _Num(self, a, allowreturn=False):

~/.local/lib/python3.5/site-packages/phylanx-0.0.1-py3.5-linux-x86_64.egg/phylan
x/ast/transformation.py in recompile(self, a, allowreturn)
    490                 return self.nodes[nm](self, a, allowreturn)
    491             else:
--> 492                 return self.nodes[nm](self, a)
    493         except NotImplementedError:
    494             raise Exception('unsupported AST node type: %s' % nm)

~/.local/lib/python3.5/site-packages/phylanx-0.0.1-py3.5-linux-x86_64.egg/phylan
x/ast/transformation.py in _Module(self, a, allowreturn)
    273     def _Module(self, a, allowreturn=False):
    274         args = [arg for arg in ast.iter_child_nodes(a)]
--> 275         return self.recompile(args[0])
    276
    277     def _Return(self, a, allowreturn=False):

~/.local/lib/python3.5/site-packages/phylanx-0.0.1-py3.5-linux-x86_64.egg/phylan
x/ast/transformation.py in recompile(self, a, allowreturn)
    490                 return self.nodes[nm](self, a, allowreturn)
    491             else:
--> 492                 return self.nodes[nm](self, a)
    493         except NotImplementedError:
    494             raise Exception('unsupported AST node type: %s' % nm)

~/.local/lib/python3.5/site-packages/phylanx-0.0.1-py3.5-linux-x86_64.egg/phylan
x/ast/transformation.py in _FunctionDef(self, a, allowreturn)
    209                     s += self.recompile(aa, True)
    210                 else:
--> 211                     s += self.recompile(aa, False)
    212                     s += ", "
    213             s += ")"

~/.local/lib/python3.5/site-packages/phylanx-0.0.1-py3.5-linux-x86_64.egg/phylan
x/ast/transformation.py in recompile(self, a, allowreturn)
    490                 return self.nodes[nm](self, a, allowreturn)
    491             else:
--> 492                 return self.nodes[nm](self, a)
    493         except NotImplementedError:
    494             raise Exception('unsupported AST node type: %s' % nm)
```

In [4]:
```
@Phylanx()
def fib2(n):
    if n < 2:
        return n
    else:
        return fib2(n-1)+fib2(n-2) # OK, if every switch in an if/elif/else block
                                   # does a return, that counts as a single return
```

In [5]:
```
fib2(8)
```

Out[5]:
```
21.0
```

In [6]:
```
@Phylanx(debug=True)
def sw(s):
    # if / elif /else works
    if s == "hello":
        return "hi"
    elif s == "goodbye":
        return "bye"
    else:
        return "?"
print(sw("goodbye"))
```

```
define(
    sw,
    s,
    if(
        (s == "hello"),
        "hi",
        if(
            (s == "goodbye"),
            "bye",
            "?"
        )
    )
)
bye
```

In [7]:
```
@Phylanx(debug=True)
def floop():
    sum = 0
    n = -1
    for i in range(10,1,n): # Basic for loops work
        sum += i
    return sum
print(floop())
```

```
define(
    floop,
    block(
        define(sum, 0),
        define(
            n,
            -(1)
        ),
        if(
            n>0,
            for(
                define(i, 10),
                i<1,
                store(i, i+n),
                block(
                    store(sum, sum+i)
                )
            ),
            for(
                define(i, 10),
                i>1,
                store(i, i+n),
                block(
                    store(sum, sum+i)
                )
            )
        ),
        sum
    )
)
54.0
```

In [13]:
```
@Phylanx()
def wloop():
    sum = 0
    i = 0
    while i < 10: # basic while loops work
        sum += i
        i += 1
    return sum
print(wloop())
```

```
45.0
```

In [14]:
```
import numpy as np

@Phylanx()
def ar():
    return constant(0,10) # creation of phylanx arrays
print(ar())
assert np.array_equal(ar(), np.zeros(10)) # compare with numpy
```

```
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
```

In [8]:
```
# %load ./examples/algorithms/phylanx_lra_csv.py
#  Copyright (c) 2018 Steven R. Brandt
#  Copyright (c) 2018 R. Tohid
#
#  Distributed under the Boost Software License, Version 1.0. (See accompanying
#  file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt)

import phylanx
from phylanx.ast import *


@Phylanx("PhySL")
def lra(file_name, xlo1, xhi1, ylo1, yhi1, xlo2, xhi2, ylo2, yhi2, alpha,
        iterations, enable_output):
    data = file_read_csv(file_name)
    x = data[xlo1:xhi1, ylo1:yhi1]
    y = data[xlo2:xhi2, ylo2]
    weights = constant(0.0, shape(x, 1))
    transx = transpose(x)
    pred = constant(0.0, shape(x, 0))
    error = constant(0.0, shape(x, 0))
    gradient = constant(0.0, shape(x, 1))
    step = 0
    while step < iterations:
        if enable_output:
            print("step: ", step, ", ", weights)
        pred = 1.0 / (1.0 + exp(-dot(x, weights)))
        error = pred - y
        gradient = dot(transx, error)
        weights = weights - (alpha * gradient)
        step += 1
    return weights


res = lra("breast_cancer.csv", 0, 569, 0, 30, 0, 569, 30, 31, 1e-5, 750, 0)
print(res)
```

```
[  1.46607779e+00   2.17539732e+00   8.60750601e+00   3.86834273e+00
   1.41035564e-02  -3.49043560e-03  -2.31089080e-02  -9.96550405e-03
   2.67907348e-02   1.10343050e-02   5.70457098e-03   1.46540259e-01
  -3.15190401e-02  -3.74357837e+00   8.18076705e-04  -1.09256181e-03
  -2.45136521e-03  -2.00536391e-04   2.48979879e-03   2.20045473e-04
   1.54975320e+00   2.79581868e+00   8.78353722e+00  -5.05096263e+00
   1.80300820e-02  -1.81653486e-02  -4.51929408e-02  -1.01027903e-02
   3.78645201e-02   1.08680459e-02]
```

In [17]:
```
# %load ./examples/algorithms/phylanx_lra_csv_np.py
#  Copyright (c) 2018 Steven R. Brandt
#  Copyright (c) 2018 R. Tohid
#
#  Distributed under the Boost Software License, Version 1.0. (See accompanying
#  file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1_0.txt)
import phylanx
import numpy as np
from phylanx.ast import *


@Phylanx("PhySL")
def lra(x, y, alpha, iterations, enable_output):
    weights = constant(0.0, shape(x, 1))
    transx = np.transpose(x)
    pred = constant(0.0, shape(x, 0))
    error = constant(0.0, shape(x, 0))
    gradient = constant(0.0, shape(x, 1))
    step = 0
    while step < iterations:
        if (enable_output):
            print("step: ", step, ", ", weights)
        pred = 1.0 / (1.0 + np.exp(-np.dot(x, weights)))
        error = pred - y
        gradient = np.dot(transx, error)
        weights = weights - (alpha * gradient)
        step += 1
    return weights


file_name = "breast_cancer.csv"

data = np.genfromtxt(file_name, skip_header=1, delimiter=",")
x = data[:, :-1]
y = data[:, -1:]
y = np.reshape(y, (y.shape[0], ))
res = lra(x, y, 1e-5, 750, 0)
print(res)
```

```
step= 0 iterations= 750
res= [  1.46607779e+00   2.17539732e+00   8.60750601e+00   3.86834273e+00
   1.41035564e-02  -3.49043560e-03  -2.31089080e-02  -9.96550405e-03
   2.67907348e-02   1.10343050e-02   5.70457098e-03   1.46540259e-01
  -3.15190401e-02  -3.74357837e+00   8.18076705e-04  -1.09256181e-03
  -2.45136521e-03  -2.00536391e-04   2.48979879e-03   2.20045473e-04
   1.54975320e+00   2.79581868e+00   8.78353722e+00  -5.05096263e+00
   1.80300820e-02  -1.81653486e-02  -4.51929408e-02  -1.01027903e-02
   3.78645201e-02   1.08680459e-02]
```

Partial list of things that don't work yet.

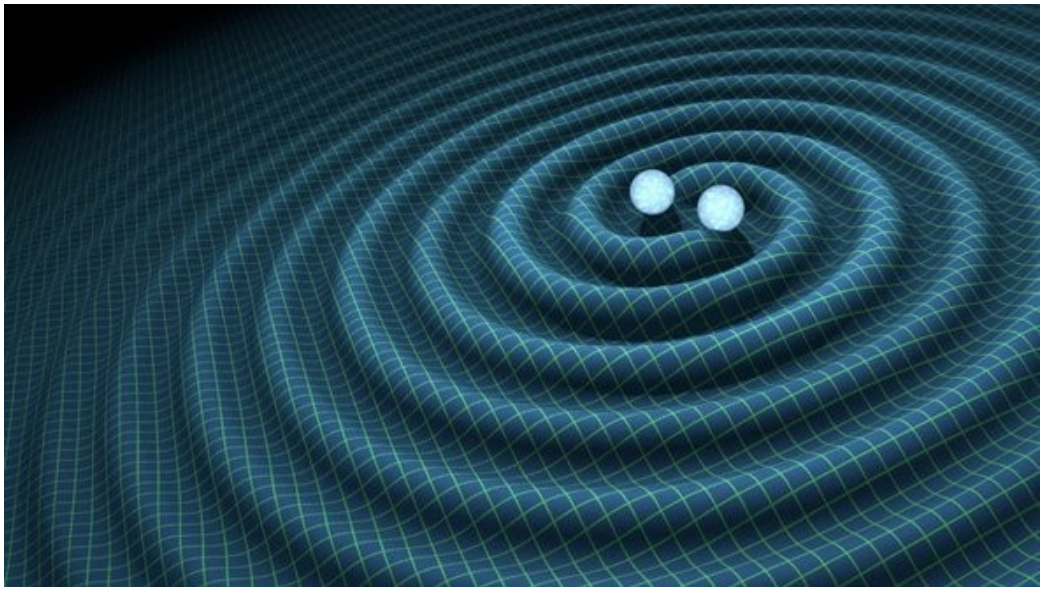
- Jupyter magics for phylanx
- parallel blocks / asyncs lambdas

</pre>

In [9]:
```
%%phylanx_cell
print("No function!")
for i in range(10):
    print("i=",i+1)
```

```
No function!
i= 1
i= 2
i= 3
i= 4
i= 5
i= 6
i= 7
i= 8
i= 9
i= 10
```

Thoughts about the future of Phylanx....



- OpenSCoP brings polyhedral calculations in the domain of Phylanx
- Stencil calculations (like black hole simulations) are in this space
- Add distributed arrays with
  - halo exchange
  - 3D Arrays
  - time integrators
  - interpolators
  - elliptic solvers?
  - AMR