Charm4Py
Productive Parallel Programming

- Recent interest in productivity in parallel programming
  - Especially as compute nodes become more heterogeneous
    - Kokkos, Raja, DPC++, …
Python

- Massive success in data science, ML
- Recent attention by HPC community

- Productive language to glue together hot code paths
  - Written in C/C++, JIT-compiled to heterogeneous devices
Charm4Py

- Productivity + performance:
  - Distributed execution of Charm++ with rich software ecosystem of Python

- pandas
- SciPy
- Numba
- NumPy
- TensorFlow
Charm4Py Basics

- Define chare classes in Python
- Create a main driver that creates chares, begins execution.
  - That’s it!
- No opaque compile errors, no additional files required
from charm4py import Chare, Array, charm

class HelloWorld(Chare):
    def sayHello(self):
        print("Hello from char: ", self.thisIndex[0],
              "on PE", charm.myPe() )

def main(args):
    chares = Array(HelloWorld, 8)
    chares.sayHello()
    charm.start(main)
from charm4py import Chare, Array, charm

class HelloWorld(Chare):
    def sayHello(self):
        print("Hello from chare: ", self.thisIndex[0], " on PE ", charm.myPe())

def main(args):
    shares = Array(HelloWorld, 8)
    shares.sayHello()
charm.start(main)
Overview

- Numpy, Pandas, Scipy
- User Code
- Runtime
- Python/C Interaction
- Charm++ Runtime
- Numba, PyCUDA
Charm4Py: Fibonacci

```python
from charm4py import charm, Chare, Future

class Fib(Chare):
    def __init__(self, n, parent, isroot):
        self.parent = parent
        self.count = 2
        self.total = 0
        self.isroot = isroot
        self.n = n

        if n <= 1:
            self.respond(n)
            return

        Chare(Fib, args=[n-1, self.thisProxy, False])
        Chare(Fib, args=[n-2, self.thisProxy, False])
```
Charm4Py: Fibonacci

```python
from charm4py import charm, Chare, Future

class Fib(Chare):
    def __init__(self, n, parent, isroot):
        self.parent = parent
        self.count = 2
        self.total = 0
        self.isroot = isroot
        self.n = n

    def main(args):
        done_future = Future()
        Chare(Fib, args=[int(args[1]), done_future, True])
        fn = done_future.get()
        print(f"Fib({int(args[1])}) = {fn}")
        charm.exit()
```

Computer Science
from charm4py import charm, Chare, Future

class Fib(Chare):
    def __init__(self, n, parent, isroot):
        self.parent = parent
        self.count = 2
        self.total = 0
        self.isroot = isroot
        self.n = n

        if n <= 1:
            self.respond(n)
            return

    Chare(Fib, args=[n-1, self.thisProxy, False])
    Chare(Fib, args=[n-2, self.thisProxy, False])

def result(self, val):
    self.total += val
    self.count -= 1
    if self.count == 0:
        self.respond(self.total)

    def respond(self, total):
        if self.isroot:
            self.parent.send(total)
        else:
            self.parent.result(total)
class Cell(Chare):
    
def __init__(self, array_dims, max_particles_per_cell_start, sim_done_future):
    # store future to notify main function when simulation is done
    self.sim_done_future = sim_done_future
    self.iteration = 0
    cellsize = (SIM_BOX_SIZE / array_dims[0], SIM_BOX_SIZE / array_dims[1])
    self.cells size = cellsize
class Cell(Chare):

def __init__(self, array_dims, max_particles_per_cell_start, sim_done_future):
    # store future to notify main function when simulation is done
    self.sim_done_future = sim_done_future
    self.iteration = 0
    cells = Array(Cell, (chares_x, chares_y),
        args=[(chares_x, chares_y), max_particles_per_cell_start, sim_done],
        useAtSync=True)

    sim_done = Future()
if self.iteration % 10 == 0:
    self.reduce(self.thisProxy[(0,0)].reportMax, len(self.particles), Reducer.max)

if self.iteration % 20 == 0:
    self.AtSync()
    self.iteration += 1
return
Arrays: Reductions, Load Balancing

```python
if self.iteration % 10 == 0:
    self.reduce(self.thisProxy[(0,0)].reportMax, len(self.particles), Reducer.max)

if self.iteration % 20 == 0:
    self.AtSync()
    self.iteration += 1
return

def resumeFromSync(self):
    self.thisProxy[self.thisIndex].run()
```
Arrays: Reductions, Load Balancing

```python
if self.iteration % 10 == 0:
    self.reduce(self.thisProxy[(0,0)].reportMax, len(self.particles), Reducer.max)

if self.iteration % 20 == 0:
    self.AtSync()
    self.iteration += 1
    return

def resumeFromSync(self):
    self.thisProxy[self.thisIndex].run()
```

No need to define a PUP! Pickle is used
Channels

• Point-to-point communication channel between chares
Channels

- Point-to-point communication channel between chares

```python
self.neighbor_indexes = self.getNbIndexes(array_dims)
self.neighbors = [Channel(self, remote=self.thisProxy[idx]) for idx in self.neighbor_indexes]
```
Channels

Point-to-point communication channel between chares

```python
self.neighbor_indexes = self.getNbIndexes(array_dims)
self.neighbors = [Channel(self, remote=self.thisProxy[idx]) for idx in self.neighbor_indexes]

for i, channel in enumerate(self.neighbors):
    channel.send(outgoingParticles[self.neighbor_indexes[i]])

for channel in charm.iwait(self.neighbors):
    incoming = channel.recv()
    self.particles += [Particle(float(incoming[i]),
                               float(incoming[i+1])) for i in range(0, len(incoming), 2)]
```
Reductions: Targeting Futures
Reductions: Targeting Futures

```python
self.reduce(self.sim_done_future)
```
Reductions: Targeting Futures

Main char:

```python
self.reduce(self.sim_done_future)
```

```python
print('
Starting simulation')
t0 = time.time()
cells.run()  # this is a broadcast
sim_done.get()
print('Particle simulation done, elapsed time=\', round(time.time() - t0, 3), 'secs')
exit()
```
Running Programs

On your local machine
Running Programs

On your local machine

```bash
python3 -m charmrun.start +p2 ./fib.py 10
```
Running Programs

On your local machine

```
python3 -m charmrun.start +p2 ./fib.py 10
```

On a cluster
Running Programs

On your local machine

```
python3 -m charmrun.start +p2 ./fib.py 10
```

On a cluster

```
python3 -m charmrun.start +p2 ./fib.py 10
mpirun -np 2 python3 ./fib.py 10
```
More Information

Source: https://github.com/UIUC-PPL/charm4py

Docs: https://charm4py.readthedocs.io/en/latest/