charm4py: Python Parallel Programming Framework

Juan J. Galvez
University of Illinois at Urbana-Champaign

Charm++ BoF SC'18

PPL UIUC Parallel Programming Laboratory

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Charm++ with Python

- Parallel/distributed programming framework for Python, built on top of Charm++
- Simplified programming model and API
- Interactive mode for parallel/distributed applications
- Wide support for futures for direct-style programming with blocking semantics (still does computation/communication overlap internally)

Starting the Application

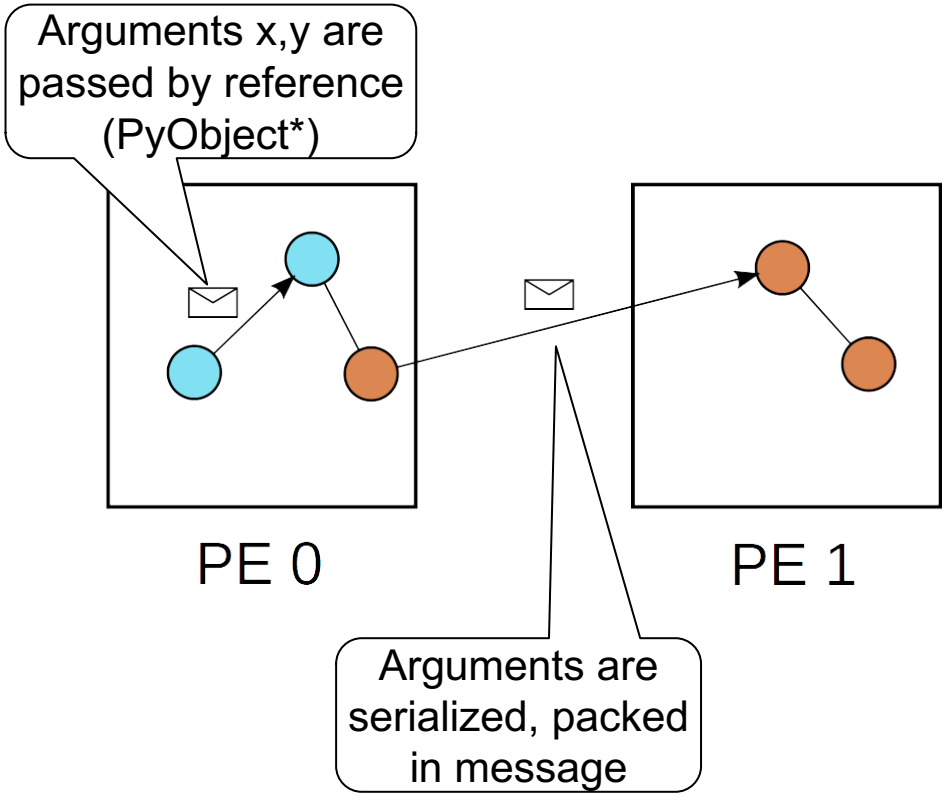
- 1) Launch multiple processes using job launcher (similar to MPI). Provided *charmrun* tool suitable for workstations and small clusters
  - python3 -m charmrun.start +p20 myprog.py
  - srun job_params /path/to/python myprog.py

Simple example

```
from charm4py import charm, Chare, Group, Array

class Test(Chare):
  def myMethod(self, x, y, z):
      print("Hello from element", self.thisIndex, x, y, z)

# program entry point, runs on one process
def main(args):
    # create and distribute work
    testGroup = Group(Test)
    testArray = Array(Test, 100)
    testGroup.myMethod([1,2,3], numpy.arange(1,100), "test_group")
    testArray.myMethod([4,5,6], numpy.arange(50,100), "test_array")

# initialize runtime
charm.start(main)
```

Remote Method Invocation

- proxy.method(x, y)
- Data arrays (e.g. numpy arrays) are directly copied into a message buffer in Charm++
- All other arguments are serialized using the *pickle* library (can automatically serialize most Python types)
- Pickling can also be customized



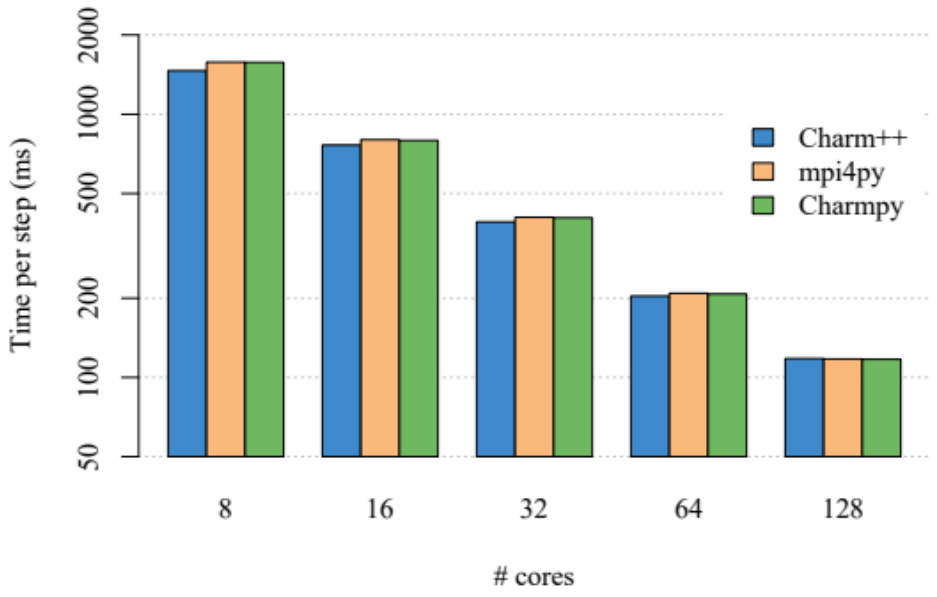
Futures

- Remote method invocation (i.e. via proxy) is **asynchronous**, returns immediately
- Can obtain a *future*:

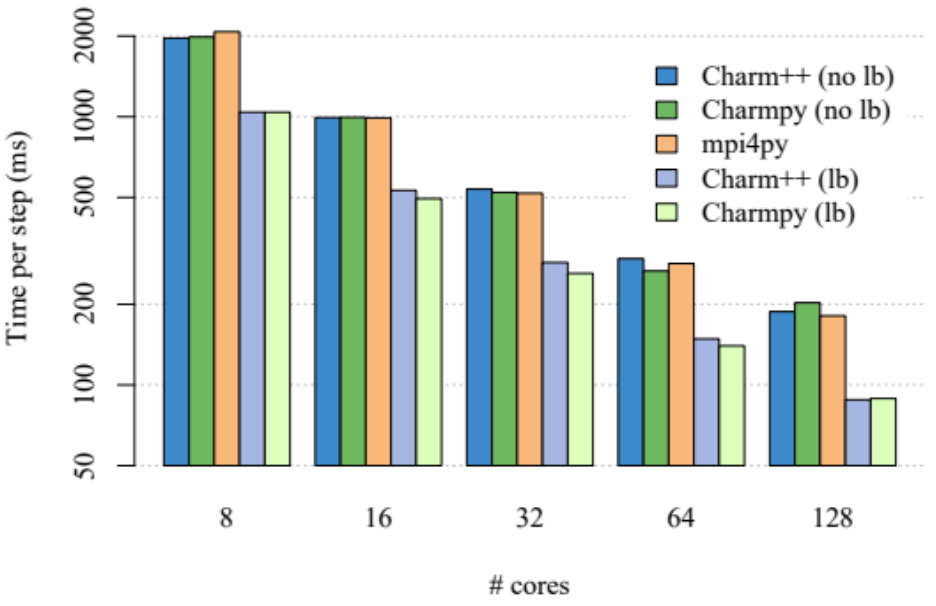
```
f1 = proxy.method1(*args, ret=True)
f2 = proxy.method2(*args, ret=True)
f3 = proxy.method3(*args, ret=True)
... do work ...
# wait for values now
vals = [f1.get(), f2.get(), f3.get()]
```

- get() blocks until result arrives. If broadcast call, returns when all chares have executed the method
- allReduce, barrier and others also supported (in next version)

stencil3d code on Cori



2 nodes. No imbalance.



2 nodes. Synthetic imbalance.
4 blocks (chares) per process

Thank you

Resources:

https://charm4py.readthedocs.io

https://github.com/UIUC-PPL/charmpy