What is Charm4Py?

- Parallel/distributed programming framework for Python
- Charm++ programming model (Charm++ for Python)
- High-level, general purpose, simple API
- Runs on top of Charm++ runtime (C++)
- Good runtime performance
- Adaptive runtime features: asynchronous remote method invocation, dynamic load balancing, automatic communication/computation overlap

Charm4Py Python-derived benefits

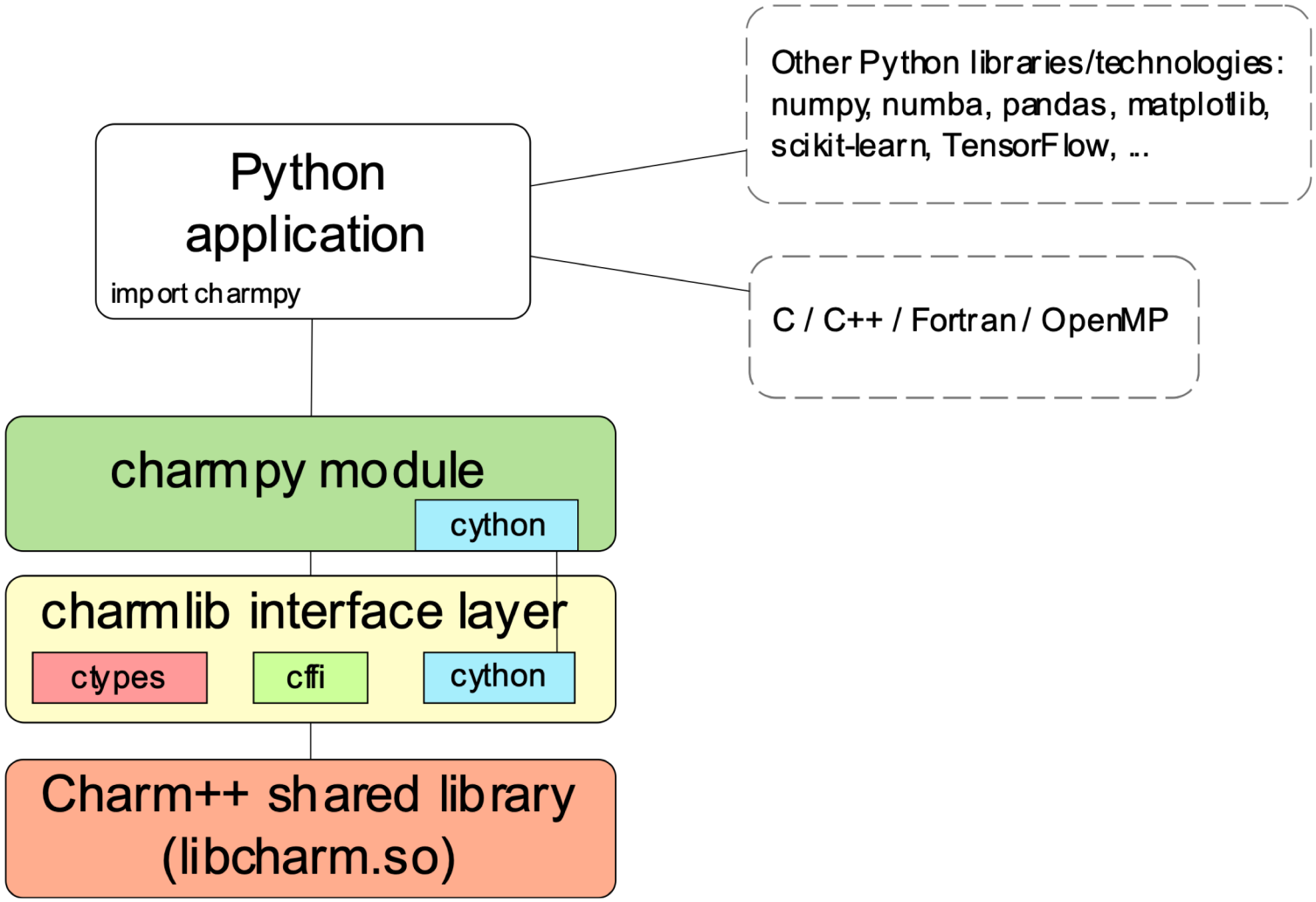
- Simplifies Charm++ programming
- Productivity (high-level, fewer lines of code, easy to debug)
- Automatic memory management
- Automatic object serialization
  - No need to define serialization (PUP) routines
  - Can customize serialization if needed
- Easy access to Python software libraries (Numba, NumPy, Pandas, scikit-learn, TensorFlow, etc)
- No .ci files!

Fibonacci example

```
from charm4py import charm, coro
@coro
def fib(n):
    if n < 2:
        return n
    else:
        return sum(charm.pool.map(fib, [n-1, n-2]))

def main(args):
    print('Result is', fib(12))
    exit()
charm.start(main)
```

Charm4Py components



What about performance?

- Many (compiled) parallel programming languages proposed over the years for HPC
- Usual Python HPC approach: high-level language driving machine-optimized compiled code
  - Numpy (high-level arrays/matrices API, native implementation)
  - Numba (JIT compiles Python “math/array” code)
  - Cython (compile generic Python to C)

Chares are distributed Python objects

- Remote methods invoked like regular Python objects, via proxy: `obj_proxy.doWork(x, y)`
- Objects are migratable (handled by Charm++ runtime)
- Method invocation asynchronous (good for performance)
- Can also do: `ret = obj_proxy.getVal(block=True)`
  - Caller gets value returned by remote method
  - Entry method on which call is made needs to be marked as @coro (runtime will inform)
  - Thread pauses but process continues scheduling messages

Distributed collections (Groups, Arrays)

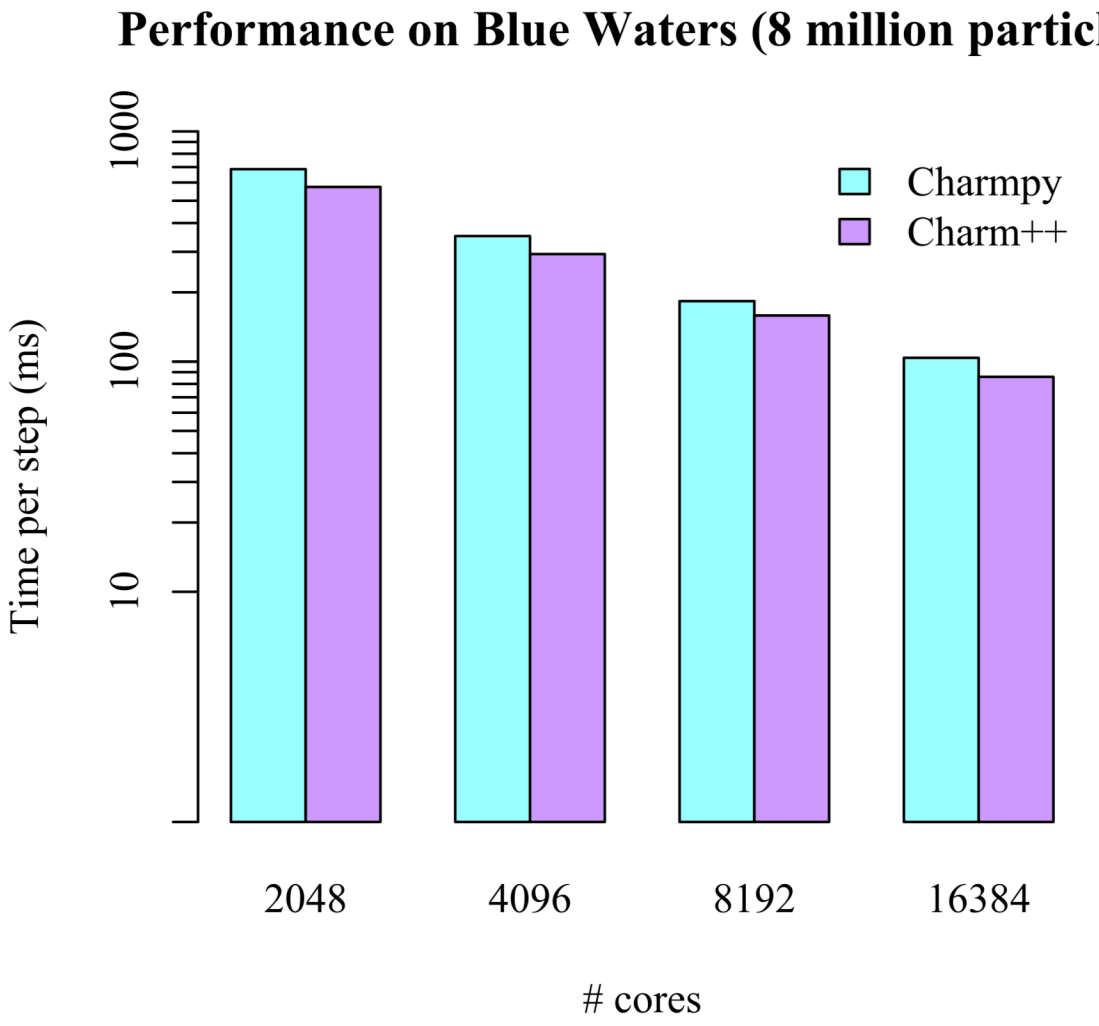
```
group = Group(MyChare)  # one instance per PE

array = Array(MyChare, (100,100))  # 2D array, 100x100 instances
array.work(x,y,z)          # invoke method on all objects in array
array[3,10].work(x,y,z) # invoke method on object with index (3,10)
```

Charm4Py 1.0 New Features

- Coroutines
  - Low overhead synchronization
- Futures
  - Flexible dependency
- Channels
  - In order pairwise communication
- Task Pool (experimental)
  - Efficient management of small granularity
  - Task style programming support
- Interactive development (experimental)
- Quiescence Detection
- Profiling

LeanMD results on Blue Waters



MD mini-app for Charm++ (http://charmplusplus.org/miniApps/#leanmd)

- Simulates the behavior of atoms based on the Lennard-Jones potential

Avg difference is 19%

(results not based on latest Charm4Py version)

More info about Charm4Py

- Documentation and tutorial at http://Charm4Py.readthedocs.io
- Examples in project repo:
  - https://github.com/UIUC-PPL/Charm4Py