Introduction
HPC Python

Antonio Gómez-Iglesias
agomez@tacc.utexas.edu

October 30th, 2014
Why Python

- Easy!
- Nice, readable code
- Great for prototyping
- Many third party libraries
Data Types

Dynamic language, but also a strongly typed language

- Objects have a type, which is determined at runtime
- A variable is a value bound to a name: the value has a type, but the variable doesn’t
- The interpreter keeps track of all variable types
- You can’t do anything that’s incompatible with the type of data you’re working with:
  - You can do ’string+string’ and it will concatenate the strings
  - You can do ’integer+integer’
  - You can’t do ’string+integer’
Data Structures

Python List

- Dynamic arrays
- Indexed structure
- Items: Python objects
- Items of different types
- Insertion and deletion at random positions
Dictionary

- Associative arrays (key-value pairs)
- Indexed by key (string or number)
- Key: unique
- Value: any Python object
- Main operation: store a value with some key and extract the value given the key
Python in HPC

• You’ll hear that Python is slow
• If it’s slow, why should you use it?
• If you already have a Python code, what should you do?
Python in Stampede

- python/2.7.3-epd-7.3.2
- python/2.7.6
- You can install your own modules:
  - python setup.py install --user
  - python setup.py install --home=<dir>
  - pip install --user module_name
- You can use virtualenv
Before We Begin

From XSEDE

ssh username@login.xsede.org
gsissh -p 2222 stampede.tacc.xsede.org

Local

ssh -Y username@stampede.tacc.utexas.edu

Python Exercises

cp ~train00/python-hpc.tar.gz .
tar -xzf python-hpc.tar.gz
module load intel/14.0.1.106
module load python/2.7.6
idev -t 2:00:00 -A TACC-HPC-PYTHON
Profiling

```python
from math import sqrt

def hello():
    print "Hello world"

def sum():
    for i in range(10000):
        a = 1
        b = 1
        c = a+b

def vector():
    a = [1., 2., 3., 4., 5., 6., 7.]*100000
    for i in a:
        t = sqrt(i**2)
        r = a.reverse()
        s = a.sort()
    print reduce(lambda x, y: x + y, a)

if __name__ == '__main__':
    hello()
    sum()
    vector()
```

Hello world

2800000.0

1400008 function calls in 0.391 seconds

Ordered by: standard name

<table>
<thead>
<tr>
<th>ncalls</th>
<th>tottime</th>
<th>percall</th>
<th>cumtime</th>
<th>percall</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename:lineno (function)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.002</td>
<td>0.002</td>
<td>0.391</td>
<td>0.391</td>
</tr>
<tr>
<td>prof.py:1(&lt;module&gt;)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.105</td>
<td>0.105</td>
<td>0.389</td>
<td>0.389</td>
</tr>
<tr>
<td>prof.py:12(vector)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>699999</td>
<td>0.061</td>
<td>0.000</td>
<td>0.061</td>
<td>0.001</td>
</tr>
<tr>
<td>prof.py:18(&lt;lambda&gt;)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>prof.py:3(hello)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.000</td>
<td>0.000</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>prof.py:6(sum)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>700000</td>
<td>0.038</td>
<td>0.000</td>
<td>0.038</td>
<td>0.038</td>
</tr>
<tr>
<td>math.sqrt</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.089</td>
<td>0.089</td>
<td>0.089</td>
<td>0.089</td>
</tr>
<tr>
<td>{method 'sort'}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>{range}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.095</td>
<td>0.095</td>
<td>0.156</td>
<td>0.156</td>
</tr>
<tr>
<td>{reduce}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
License

© The University of Texas at Austin, 2014

This work is licensed under the Creative Commons Attribution Non-Commercial 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc/3.0/

When attributing this work, please use the following text: ”HPC Python”, Texas Advanced Computing Center, 2014. Available under a Creative Commons Attribution Non-Commercial 3.0 Unported License.