Fast numerics in Python - NumPy and PyPy

Maciej Fijałkowski

SEA, NCAR

22 February 2012
What is this talk about?

- What is PyPy and why?
- Numeric landscape in Python
- What we achieved in PyPy
- Where we’re going?
What is PyPy?

- An efficient implementation of Python language
- A framework for writing efficient dynamic language implementations
- An open source project with a lot of volunteer effort, released under the MIT license
- Agile development, 13000 unit tests, continuous integration, sprints, distributed team
- I’ll talk today about the first part (mostly)
PyPy status right now

- An **efficient just in time compiler** for the Python language
- Relatively “good” on numerics (compared to other dynamic languages)
- Example - real time video processing
- 2-300x faster on Python code
Why should you care?

- *If I write this stuff in C/fortran/assembler it’ll be faster anyway*
- maybe, but ...
Experimentation is important
Implementing something faster, in human time, leaves more time for optimizations and improvements
For novel algorithms, clearer implementation makes them easier to evaluate (Python often is cleaner than C)
Sometimes makes it possible in the first place
Experimentation is important

Implementing something faster, in human time, leaves more time for optimizations and improvements

For novel algorithms, clearer implementation makes them easier to evaluate (Python often is cleaner than C)

Sometimes makes it possible in the first place
Why would you care even more?

- Growing community
- Everything is for free with reasonable licensing
- There are many smart people out there addressing hard problems
Example of why would you care

- You spend a year writing optimized algorithms for a GPU
- Next year a new generation of GPUs come along
- Your algorithms are no longer optimized

- Alternative - express your algorithms
- Leave low-level details to people who have nothing better to do

- ... like me (I don’t know enough Physics to do the other part)
Example of why would you care

- You spend a year writing optimized algorithms for a GPU
- Next year a new generation of GPUs come along
- Your algorithms are no longer optimized

- Alternative - express your algorithms
- Leave low-level details to people who have nothing better to do

- ... like me (I don’t know enough Physics to do the other part)
Example of why would you care

- You spend a year writing optimized algorithms for a GPU
- Next year a new generation of GPUs come along
- Your algorithms are no longer optimized

Alternative - express your algorithms
- Leave low-level details to people who have nothing better to do
- ... like me (I don’t know enough Physics to do the other part)
Numerics in Python

- **numpy** - for array operations
- **scipy, scikits** - various algorithms, also exposing C/fortran libraries
- **matplotlib** - pretty pictures
- **ipython**
There is an entire ecosystem!

- Which I don’t even know very well
- PyCUDA
- pandas
- mayavi
What’s important?

- There is an entire ecosystem built by people
- It’s available for free, no shady licensing
- It’s being expanded
- It’s growing
- It’ll keep up with hardware advancements
Problems with numerics in python

- Stuff is reasonably fast, but...
  - Only if you don’t actually write much Python
  - Array operations are fine as long as they’re vectorized
  - Not everything is expressable that way
  - Numpy allocates intermediates for each operation, suboptimal
Problems with numerics in python

- Stuff is reasonably fast, but...
- Only if you don’t actually write much **Python**
- Array operations are fine as long as they’re vectorized
- Not everything is expressable that way
- Numpy allocates intermediates for each operation, suboptimal
Our approach

- Build a tree of operations
- Compile assembler specialized for aliasing and operations
- Execute the specialized assembler
Examples

- $a, b, c$ are single dimensional arrays
- $a + a$ would generate different code than $a + b$
- $a + b \times c$ is as fast as a loop
## Performance comparison

<table>
<thead>
<tr>
<th>Expression</th>
<th>NumPy</th>
<th>PyPy</th>
<th>PyPy</th>
<th>GCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a+b)</td>
<td>0.6s</td>
<td>0.4s</td>
<td>0.3s</td>
<td>(0.25s)</td>
</tr>
<tr>
<td>(a+b+c)</td>
<td>1.9s</td>
<td>0.5s</td>
<td>0.7s</td>
<td>(0.32s)</td>
</tr>
<tr>
<td>(5+)</td>
<td>3.2s</td>
<td>0.8s</td>
<td>1.7s</td>
<td>(0.51s)</td>
</tr>
</tbody>
</table>

- Pathscale is actually slower
## Performance comparison SSE

- Branch only so far!

<table>
<thead>
<tr>
<th></th>
<th>PyPy SSE</th>
<th>PyPy</th>
<th>GCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>a+b</td>
<td>0.3s</td>
<td>0.4s</td>
<td>0.3s (0.25s)</td>
</tr>
<tr>
<td>a+b+c</td>
<td>0.35s</td>
<td>0.5s</td>
<td>0.7s (0.32s)</td>
</tr>
<tr>
<td>5+</td>
<td>0.36s</td>
<td>0.8s</td>
<td>1.7s (0.51s)</td>
</tr>
</tbody>
</table>
This works reasonably well
Far from implementing the entire numpy, although it’s in progress
Assembler generation backend needs works
Vectorization in progress
Status benchmarks - slightly more complex

- Laplace solution

  - NumPy: 4.3s
    - looped: too long to run ~2100s
  - PyPy: 1.6s
    - looped: 2.5s
  - C: 0.9s
Progress plan

- Express operations in high-level languages
- Let us deal with low level details
- However, retain knobs and buttons for advanced users
- Don’t get penalized too much for not using them
Progress plan

- Express operations in high-level languages
- Let us deal with low level details
- However, retain knobs and buttons for advanced users
- Don’t get penalized too much for not using them
Few words about the future

- Predictions are hard
  - Especially when it comes to future
  - Take this with a grain of salt
Few words about the future

- Predictions are hard
- Especially when it comes to future
- Take this with a grain of salt
PyPy is an easy platform to experiment with
We did not spend a whole lot of time dealing with the low-level optimizations
Automatic vectorization over multiple threads
SSE, GPU, dynamic offloading
Optimizations based on machine cache size
We’re running a fundraiser, make your employer donate money
http://pypy.org/
http://buildbot.pypy.org/
numpy-status/latest.html
http://morepypy.blogspot.com/

Any questions?