PROFILING, OPTIMIZATION, PARALLELIZATION

CHRISTOPH PAULIK

OPTIMIZATION

> *We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil.*
>
> *– Donald Knuth*

The problem with premature optimization is that you never know in advance where the bottlenecks will be.

WHEN TO OPTIMIZE

- Make it work.
- Make it right
- Make everything work.
- Make everything right.
- Profile Before Optimizing.
- Make it fast. You maintained unit tests, right? Then you can refactor the code mercilessly in order to improve the performance.

Guillermo Schwarz

WHAT IS GENERALLY SLOW IN PYTHON

- CPU bound tasks
- e.g. Nested for loops over large arrays
- But this does not mean that this is the reason why your program is slow
- Can be quite surprising at times

AMDAHL'S LAW

A parallelized program can only be as fast as the slowest single threaded piece of code.

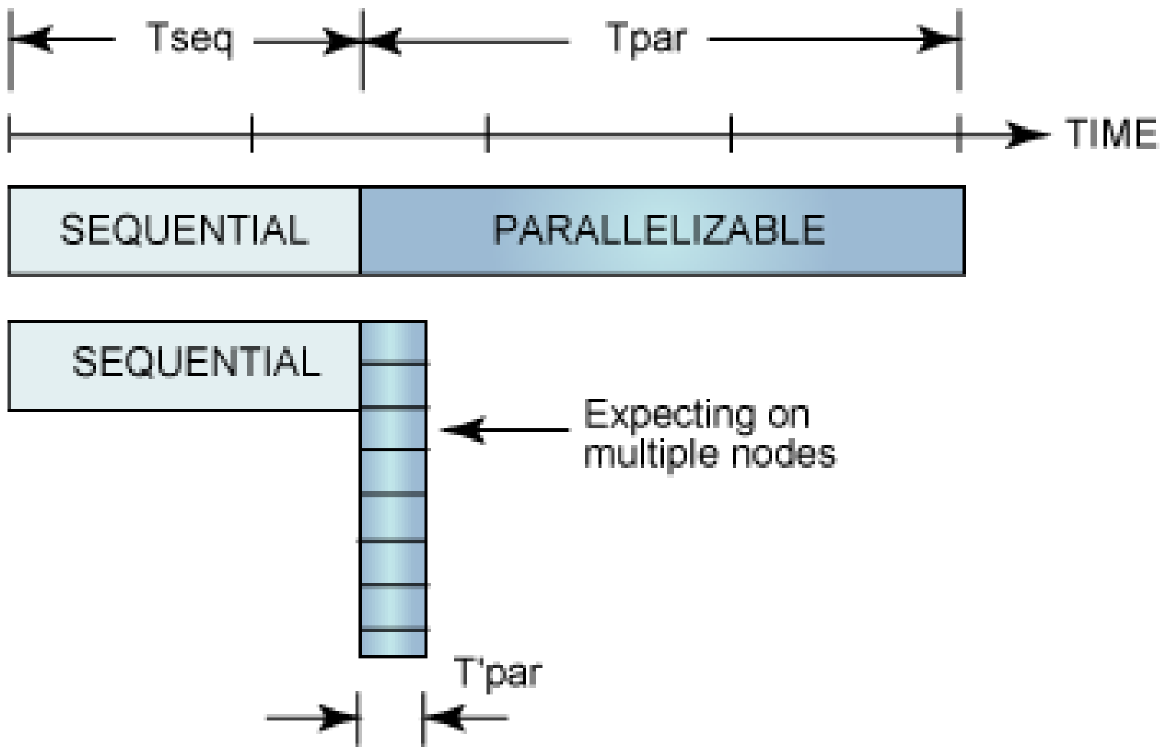


Figure 1: Illustration of Amdahl's law from IBM

PROFILING

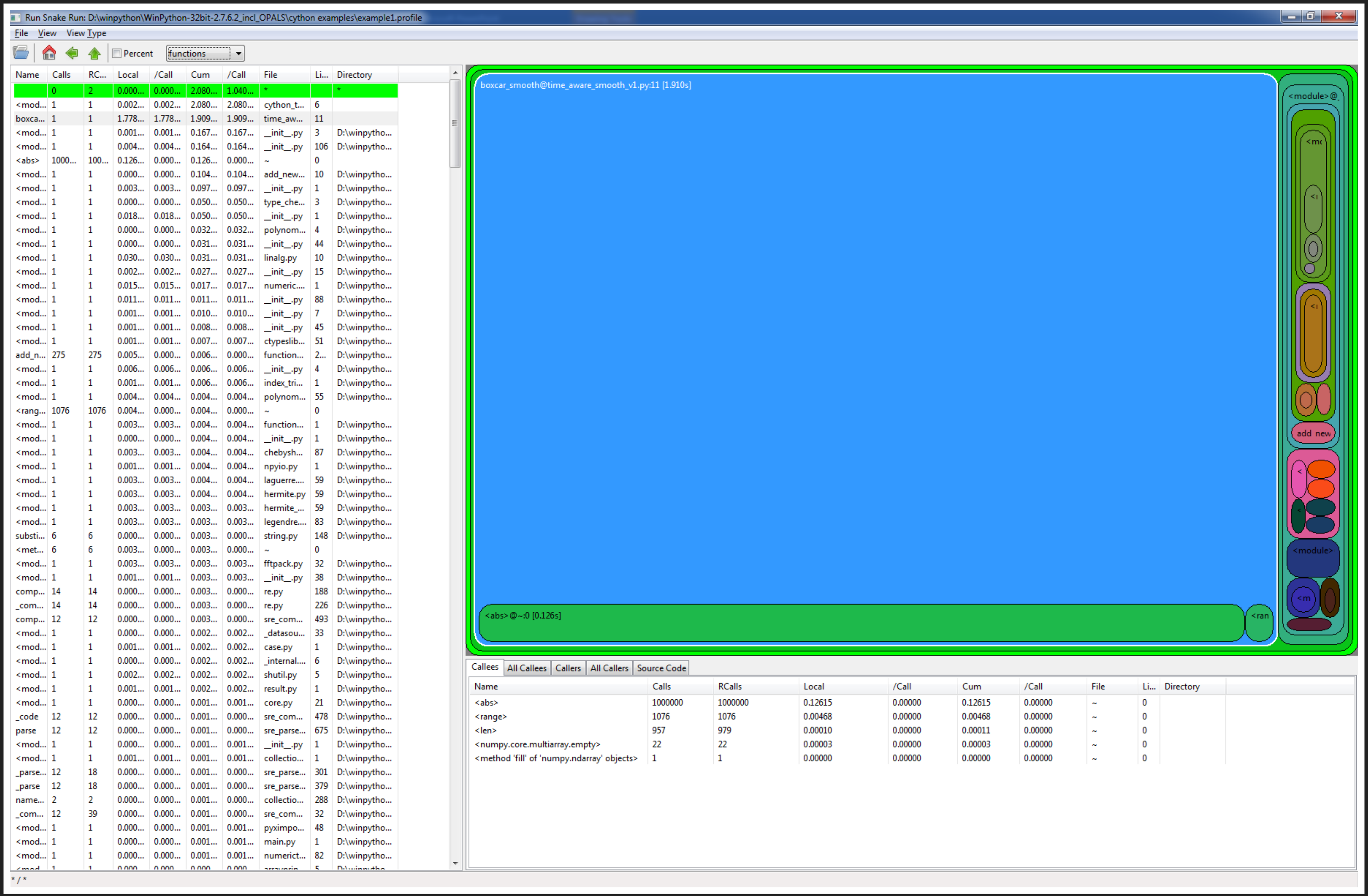
- Built in Python profiler can be called from command line or from Python itself or inside a IPython Notebook.

```
python -m cProfile -o <output> <script-name> <options>
```

- Viewing profile

```
python –m pstats <output>
runsnake <output>
```

Runsnakerun requires wxpython – not portable


Run Snake Run: D:\winpython\WinPython-32bit-2.7.6.2_incl_OPALS\cython examples\example1.profile
File
View
View Type
□ Percent
functions
boxcar_smooth@time_aware_smooth_v1.py:11 [1.910s]
<abs>@~:0 [0.126s]
<ran
<module>@
<mo
<i
<i
add_new
<
<module>
<m
Callees
All Callees
Callers
All Callers
Source Code
* / *

Name	Calls	RC...	Local	/Call	Cum	/Call	File	Li...	Directory
	0	2	0.000...	0.000...	2.080...	1.040...	*		*
<mod...	1	1	0.002...	0.002...	2.080...	2.080...	cython_t...	6	
boxca...	1	1	1.778...	1.778...	1.909...	1.909...	time_aw...	11	
<mod...	1	1	0.001...	0.001...	0.167...	0.167...	__init__.py	3	D:\winpytho...
<mod...	1	1	0.004...	0.004...	0.164...	0.164...	__init__.py	106	D:\winpytho...
<abs>	1000...	100...	0.126...	0.000...	0.126...	0.000...	~	0	
<mod...	1	1	0.000...	0.000...	0.104...	0.104...	add_new...	10	D:\winpytho...
<mod...	1	1	0.003...	0.003...	0.097...	0.097...	__init__.py	1	D:\winpytho...
<mod...	1	1	0.000...	0.000...	0.050...	0.050...	type_che...	3	D:\winpytho...
<mod...	1	1	0.018...	0.018...	0.050...	0.050...	__init__.py	1	D:\winpytho...
<mod...	1	1	0.000...	0.000...	0.032...	0.032...	polynom...	4	D:\winpytho...
<mod...	1	1	0.000...	0.000...	0.031...	0.031...	__init__.py	44	D:\winpytho...
<mod...	1	1	0.030...	0.030...	0.031...	0.031...	linalg.py	10	D:\winpytho...
<mod...	1	1	0.002...	0.002...	0.027...	0.027...	__init__.py	15	D:\winpytho...
<mod...	1	1	0.015...	0.015...	0.017...	0.017...	numeric....	1	D:\winpytho...
<mod...	1	1	0.011...	0.011...	0.011...	0.011...	__init__.py	88	D:\winpytho...
<mod...	1	1	0.001...	0.001...	0.010...	0.010...	__init__.py	7	D:\winpytho...
<mod...	1	1	0.001...	0.001...	0.008...	0.008...	__init__.py	45	D:\winpytho...
<mod...	1	1	0.001...	0.001...	0.007...	0.007...	ctypeslib...	51	D:\winpytho...
add_n...	275	275	0.005...	0.000...	0.006...	0.000...	function...	2...	D:\winpytho...
<mod...	1	1	0.006...	0.006...	0.006...	0.006...	__init__.py	4	D:\winpytho...
<mod...	1	1	0.001...	0.001...	0.006...	0.006...	index_tri...	1	D:\winpytho...
<mod...	1	1	0.004...	0.004...	0.004...	0.004...	polynom...	55	D:\winpytho...
<rang...	1076	1076	0.004...	0.000...	0.004...	0.000...	~	0	
<mod...	1	1	0.003...	0.003...	0.004...	0.004...	function...	1	D:\winpytho...
<mod...	1	1	0.000...	0.000...	0.004...	0.004...	__init__.py	1	D:\winpytho...
<mod...	1	1	0.003...	0.003...	0.004...	0.004...	chebysh...	87	D:\winpytho...
<mod...	1	1	0.001...	0.001...	0.004...	0.004...	npyio.py	1	D:\winpytho...
<mod...	1	1	0.003...	0.003...	0.004...	0.004...	laguerre....	59	D:\winpytho...
<mod...	1	1	0.003...	0.003...	0.004...	0.004...	hermite.py	59	D:\winpytho...
<mod...	1	1	0.003...	0.003...	0.003...	0.003...	hermite_...	59	D:\winpytho...
<mod...	1	1	0.003...	0.003...	0.003...	0.003...	legendre....	83	D:\winpytho...
substi...	6	6	0.000...	0.000...	0.003...	0.000...	string.py	148	D:\winpytho...
<met...	6	6	0.003...	0.000...	0.003...	0.000...	~	0	
<mod...	1	1	0.003...	0.003...	0.003...	0.003...	fftpack.py	32	D:\winpytho...
<mod...	1	1	0.001...	0.001...	0.003...	0.003...	__init__.py	38	D:\winpytho...
comp...	14	14	0.000...	0.000...	0.003...	0.000...	re.py	188	D:\winpytho...
_com...	14	14	0.000...	0.000...	0.003...	0.000...	re.py	226	D:\winpytho...
comp...	12	12	0.000...	0.000...	0.003...	0.000...	sre_com...	493	D:\winpytho...
<mod...	1	1	0.000...	0.000...	0.002...	0.002...	_datasou...	33	D:\winpytho...
<mod...	1	1	0.001...	0.001...	0.002...	0.002...	case.py	1	D:\winpytho...
<mod...	1	1	0.000...	0.000...	0.002...	0.002...	_internal....	6	D:\winpytho...
<mod...	1	1	0.002...	0.002...	0.002...	0.002...	shutil.py	5	D:\winpytho...
<mod...	1	1	0.001...	0.001...	0.002...	0.002...	result.py	1	D:\winpytho...
<mod...	1	1	0.000...	0.000...	0.001...	0.001...	core.py	21	D:\winpytho...
_code	12	12	0.000...	0.000...	0.001...	0.000...	sre_com...	478	D:\winpytho...
parse	12	12	0.000...	0.000...	0.001...	0.000...	sre_parse...	675	D:\winpytho...
<mod...	1	1	0.000...	0.000...	0.001...	0.001...	__init__.py	1	D:\winpytho...
<mod...	1	1	0.001...	0.001...	0.001...	0.001...	collectio...	1	D:\winpytho...
_parse...	12	18	0.000...	0.000...	0.001...	0.000...	sre_parse...	301	D:\winpytho...
_parse	12	18	0.000...	0.000...	0.001...	0.000...	sre_parse...	379	D:\winpytho...
name...	2	2	0.000...	0.000...	0.001...	0.000...	collectio...	288	D:\winpytho...
_com...	12	39	0.000...	0.000...	0.001...	0.000...	sre_com...	32	D:\winpytho...
<mod...	1	1	0.000...	0.000...	0.001...	0.001...	pyximpo...	48	D:\winpytho...
<mod...	1	1	0.000...	0.000...	0.001...	0.001...	main.py	1	D:\winpytho...
<mod...	1	1	0.000...	0.000...	0.001...	0.001...	numerict...	82	D:\winpytho...
<mod...	1	1	0.000...	0.000...	0.000...	0.000...	arrayprin...	5	D:\winpytho...

Name	Calls	RCalls	Local	/Call	Cum	/Call	File	Li...	Directory
<abs>	1000000	1000000	0.12615	0.00000	0.12615	0.00000	~	0	
<range>	1076	1076	0.00468	0.00000	0.00468	0.00000	~	0	
<len>	957	979	0.00010	0.00000	0.00011	0.00000	~	0	
<numpy.core.multiarray.empty>	22	22	0.00003	0.00000	0.00003	0.00000	~	0	
<method 'fill' of 'numpy.ndarray' objects>	1	1	0.00000	0.00000	0.00000	0.00000	~	0	

Figure 2: Runsnake interface

LINE PROFILER

- `@profile` decorator for functions that we want to look at

```
kernprof –l –o <outputfilename> <script-name> <options>
python –m line_profiler <outputfilename>
```

MORE INFORMATION ABOUT PROFILING

- Python Profilers
- Line Profiler
- Tutorial
- Pstats

IMPROVING PERFORMANCE

Now that you have found the slow part, what to do?

- Can it be done in vectorized form? – numpy? – `no !`
- Are there other packages or existing libraries that do it? – `no !`
- Did you do it in an inefficient way? – `no !`
- Is it CPU bound and not I/O or memory bound? – `yes !`

If those are you answers:

- Cython is a good way to go if you do not know C or C++.
- Numba should be able to do it automatically. (use Anaconda)

CYTHON

- Cython gives you the combined power of Python and C to let you
  - write Python code that calls back and forth from and to C or C++ code natively at any point.
  - easily tune readable Python code into plain C performance by adding static type declarations. – different ways, we’ll focus on one.
  - Windows install instructions

EXAMPLE

```
def f(x):
    return x ** 2 - x

def integrate_f(a, b, N):
    s = 0
    dx = (b - a / N)
    for i in range(N):
        s += f(a + i * dx)
    return s * dx
```

CYTHON

```
def f(double x):
    return x ** 2 - x

def integrate_f(double a, double b, int N):
    cdef int i
    cdef double s, dx
    s = 0
    dx = (b - a / N)
    for i in range(N):
        s += f(a + i * dx)
    return s * dx
```

BUILDING CYTHON CODE

- save it in a `.pyx` file
- use `pyximport` instead of regular import
- run the `cython` command line program and then compile the `.c` file manually
- use IPython Notebook

More details in the documentation

EXAMPLE

- open `profiling.ipynb`
- profiling
- find slow part
- make it faster using cython
- IPython parallelization