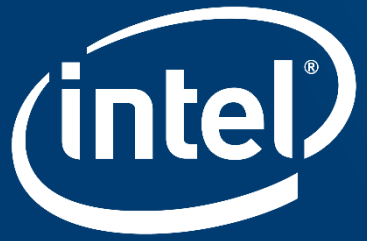


PERFORMANCE ANALYSIS OF PYTHON* APPLICATIONS WITH INTEL® VTUNE™ AMPLIFIER

Vasily Litvinov

Senior Software Engineer, Intel

Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Optimization Notice

Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Agenda

Why do we need Python optimization?

How one finds the code to tune?

Short overview of existing tools

Intel® VTune™ Amplifier capabilities and comparison

An example

Q & A

Why do we need Python optimization?

- Python is used to power wide range of software, including those where application performance matters
- Some Python code may not scale well, but you won't know it unless you give it enough workload to chew on
- Sometimes you are just not happy with the speed of your code

All in all, there are times when you want to make your code run faster, be more responsive, *(insert your favorite buzzword here)*.

So, you need to **optimize** (or tune) your code.

How one finds the code to tune – measuring vs guessing

- **Hard stare** = Often wrong

- **Logging** = Analysis is tedious

- **Profile** = Accurate, Easy

Not All Profilers Are Equal

There are different profiling techniques, for example:

- **Event-based**
  - Example: built-in Python cProfile profiler

- **Instrumentation-based**
  - Usually requires modifying the target application (source code, compiler support, etc.)
  - Example: line_profiler

- **Statistical**
  - Accurate enough & less intrusive
  - Example: vmstat, statprof

Short profilers overview

Tool	Description	Platforms	Profile level	Avg. overhead *
Intel® VTune™ Amplifier	• Rich GUI viewer • Mixed C/C++/Python code	Windows Linux	Line	~1.1-1.6x
cProfile (built-in)	• Text interactive mode: “pstats” (built-in) • GUI viewer: RunSnakeRun (Open Source) • PyCharm	Any	Function	1.3x-5x
Python Tools	• Visual Studio (2010+) • Open Source	Windows	Function	~2x
line_profiler	• Pure Python • Open Source • Text-only viewer	Any	Line	Up to 10x or more

* Measured against Grand Unified Python Benchmark

Machine specs: HP EliteBook 850 G1; Intel® Core™ i5-4300U @1.90 Ghz (4 cores with HT on) CPU; 16 GB RAM; Windows 8.1 x86_64

Mixed C/Python example to profile: **core.pyx** (Cython-based)

```
import math
cdef class SlowpokeCore:
    cdef public object N
    def __init__(self, N):
        self.N = N

    cdef double doWork(self, int N) except *:
        cdef int i, j, k
        cdef double res
        res = 0
        for j in range(N):
            k = 0
            for i in range(N):
                k += 1
            res += k
        return math.log(res)

    def __str__(self):
        return 'SlowpokeCore: %f' % self.doWork(self.N)
```

Mixed C/Python example to profile: **main.py** (Cython-based)

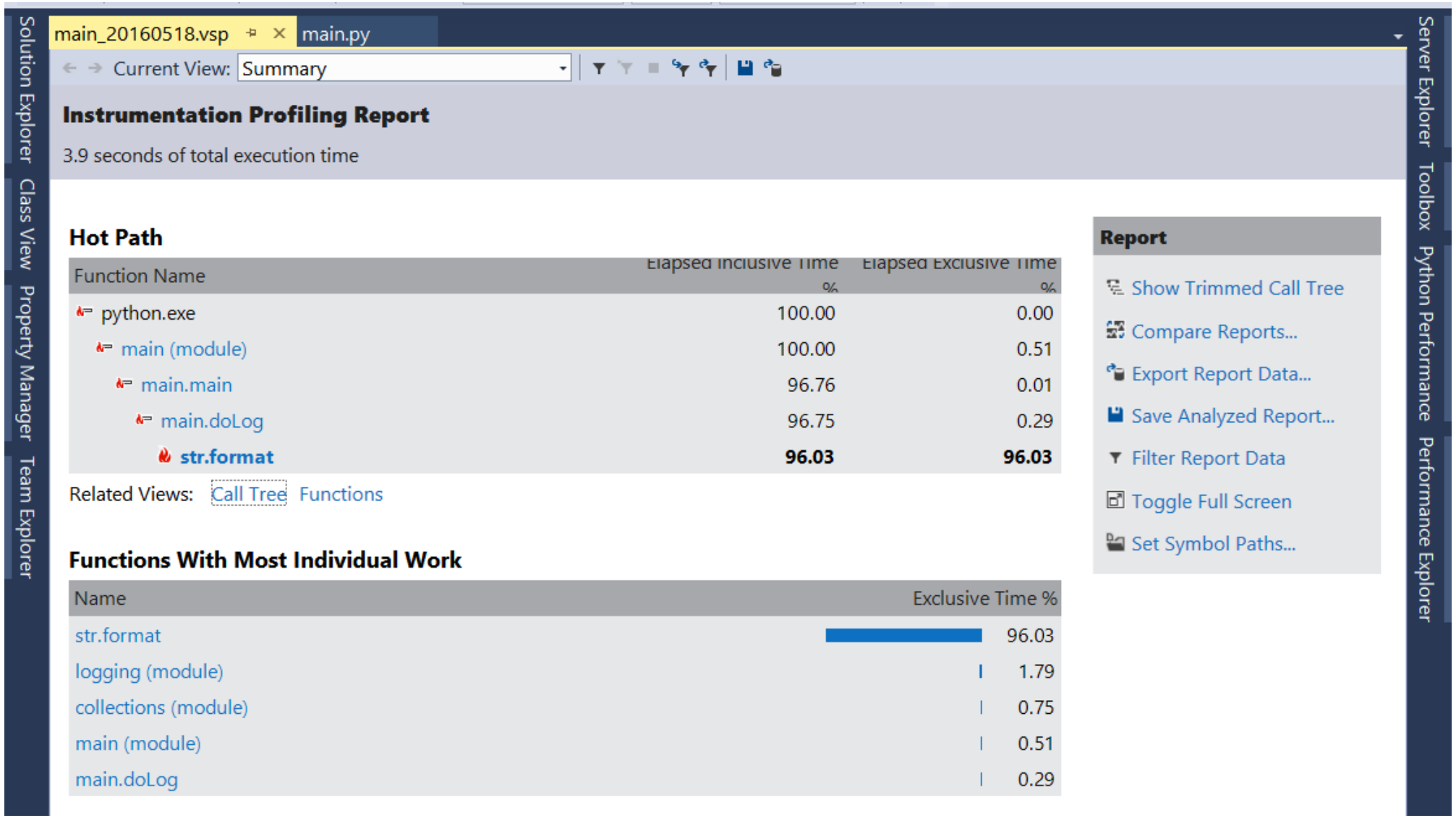
```
from slowpoke import SlowpokeCore
import logging
import time

def makeParams():
    objects = tuple(SlowpokeCore(50000) for _ in xrange(50))
    template = ''.join('{%d}' % i for i in xrange(len(objects)))
    return template, objects

def doLog():
    template, objects = makeParams()
    for _ in xrange(1000):
        logging.info(template.format(*objects))

def main():
    logging.basicConfig()
    start = time.time()
    doLog()
    stop = time.time()
    print('run took: %.3f' % (stop - start))

if __name__ == '__main__':
    main()
```

cProfile + pstats UI example

```
> python -m cProfile -o run.prof main.py
> python -m pstats run.prof

run.prof% sort cumulative
run.prof% stats 10
Wed May 18 13:15:23 2016    run.prof

         6160 function calls in 3.043 seconds

   Ordered by: cumulative time
   List reduced from 40 to 10 due to restriction <10>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    3.043    3.043 <string>:1(<module>)
        1    0.000    0.000    3.043    3.043 C:\Vass\Work\pytrace_tpss\video\main.py:15(main)
        1    0.005    0.005    3.042    3.042 C:\Vass\Work\pytrace_tpss\video\main.py:10(doLog)
     1000    3.029    0.003    3.029    0.003 {method 'format' of 'str' objects}
     1000    0.004    0.000    0.008    0.000 C:\Python27\lib\logging\__init__.py:1586(info)
     1000    0.001    0.000    0.004    0.000 C:\Python27\lib\logging\__init__.py:1122(info)
     1000    0.001    0.000    0.003    0.000 C:\Python27\lib\logging\__init__.py:1324(isEnabledFor)
     1000    0.001    0.000    0.001    0.000 C:\Python27\lib\logging\__init__.py:1310(getEffectiveLevel)
     1002    0.000    0.000    0.000    0.000 {len}
        1    0.000    0.000    0.000    0.000 C:\Vass\Work\pytrace_tpss\video\main.py:5(makeParams)
```

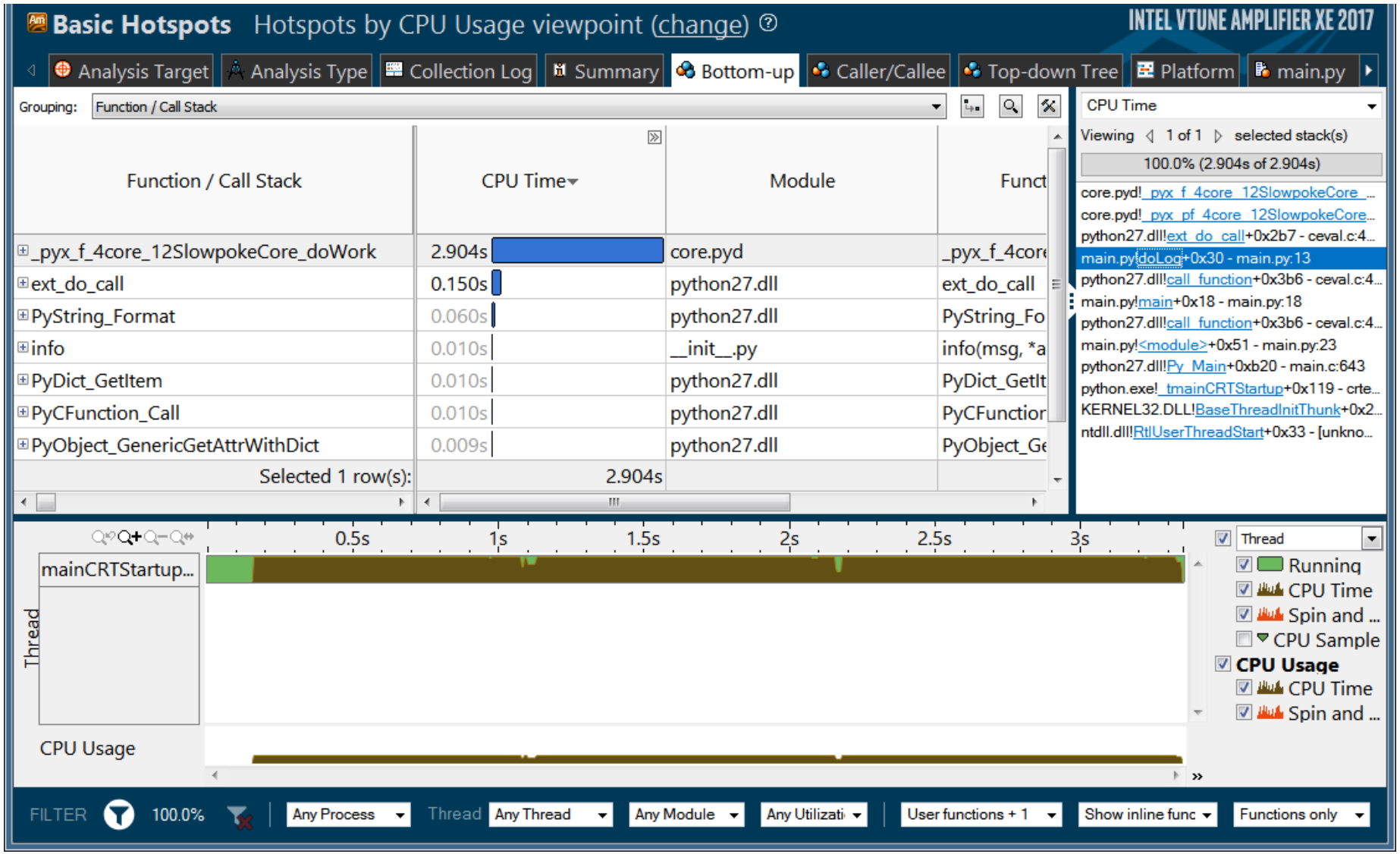

Machine specs: HP EliteBook 850 G1; Intel® Core™ i5-4300U @1.90 Ghz (4 cores with HT on) CPU; 16 GB RAM; Windows 8.1 x86_64

Python Tools GUI example



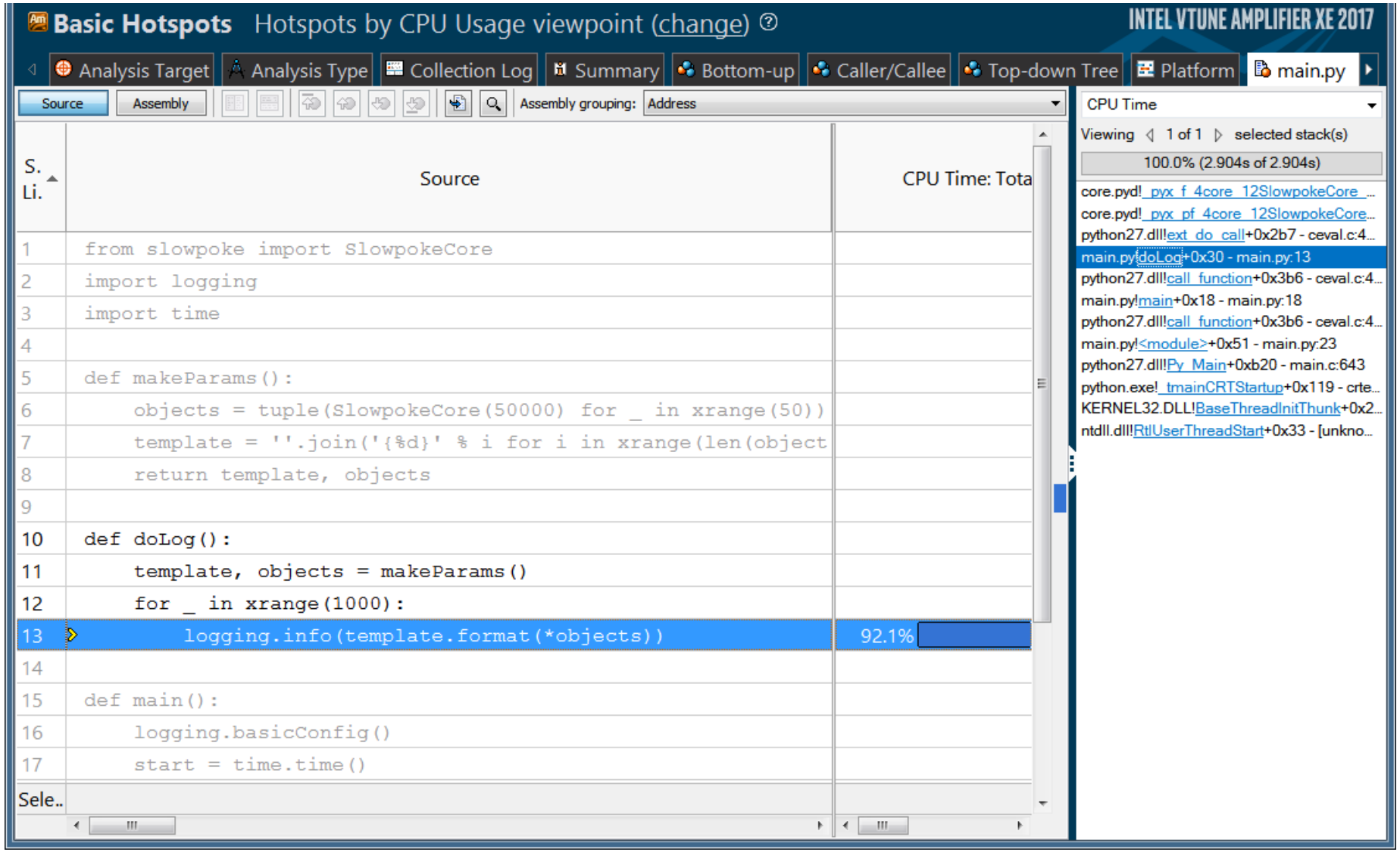
Machine specs: HP EliteBook 850 G1; Intel® Core™ i5-4300U @1.90 Ghz (4 cores with HT on) CPU; 16 GB RAM; Windows 8.1 x86_64

Intel® VTune™ Amplifier example



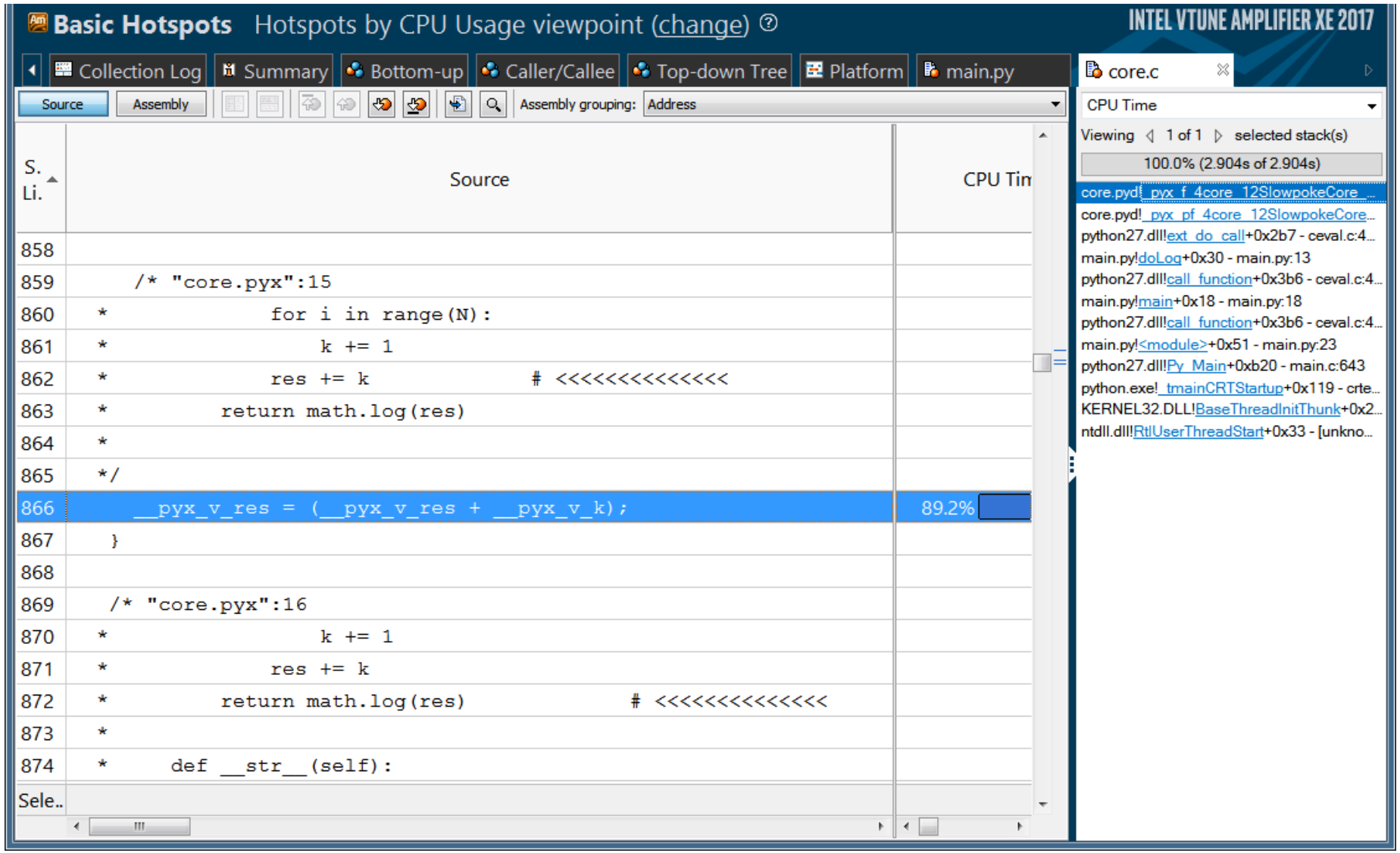
Machine specs: HP EliteBook 850 G1; Intel® Core™ i5-4300U @1.90 Ghz (4 cores with HT on) CPU; 16 GB RAM; Windows 8.1 x86_64

Intel® VTune™ Amplifier – source view (main.py)



Machine specs: HP EliteBook 850 G1; Intel® Core™ i5-4300U @1.90 Ghz (4 cores with HT on) CPU; 16 GB RAM; Windows 8.1 x86_64

Intel® VTune™ Amplifier – source view (core.c)



Machine specs: HP EliteBook 850 G1; Intel® Core™ i5-4300U @1.90 Ghz (4 cores with HT on) CPU; 16 GB RAM; Windows 8.1 x86_64

Intel® VTune™ Amplifier: Accurate & Easy

Line-level profiling details:

- Uses sampling profiling technique
- Average overhead ~1.1x-1.6x (on certain benchmarks *)

Cross-platform:

- Windows and Linux
- Python 32- and 64-bit; 2.7.x, 3.4.x, 3.5.0 versions

Rich Graphical UI

Supported workflows:

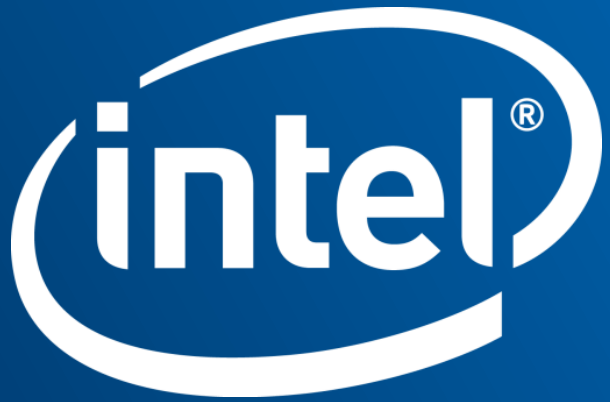
- Start application, wait for it to finish
- Attach to application, profile for a bit, detach

* Measured against Grand Unified Python Benchmark

Machine specs: HP EliteBook 850 G1; Intel® Core™ i5-4300U @1.90 Ghz (4 cores with HT on) CPU; 16 GB RAM; Windows 8.1 x86_64

Resources

- Get access to Intel® VTune™ Amplifier 2017 Beta:
  - Visit https://software.intel.com/en-us/python-profiling for registration, download and support
  - Documentation available when installed
- Get Intel® Distribution for Python*:
  - Visit https://software.intel.com/en-us/python-distribution for download, documentation and support
  - Also available at Intel channel at Anaconda (you can use "`conda install`"!): https://anaconda.org/intel
- More Intel products for developers on Intel Developer Zone
  - Visit https://software.intel.com/en-us/intel-sdp-home/ webpage


intel