Pontificia Universidad JAVERIANA Cali

Facultad de Ingeniería y Ciencias
Departamento de Electrónica y Ciencias de la Computación

Carrera de Ingeniería de Sistemas y Computación

Programación Paralela

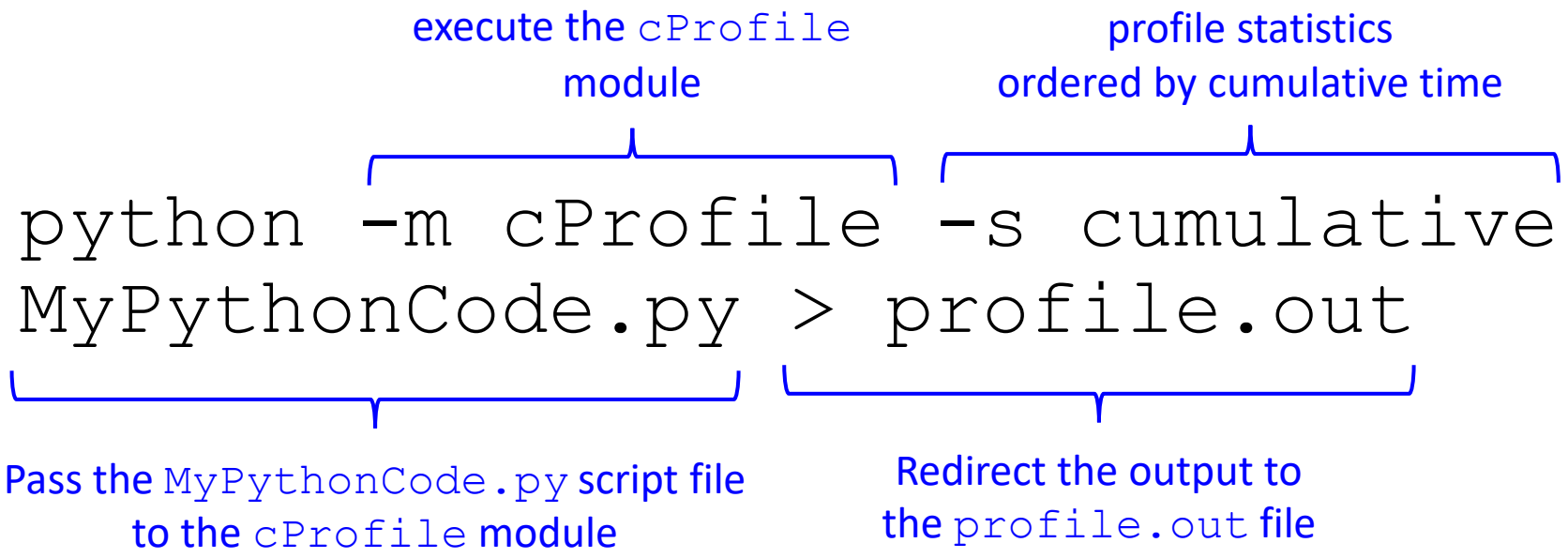
Profiling Code: Detecting Performance Bottlenecks

Mayo 2022

Tek

Profiling Code

- Using Python's built-in profiling module: The `cProfile` module (Python's recommended module to gather profiling information)



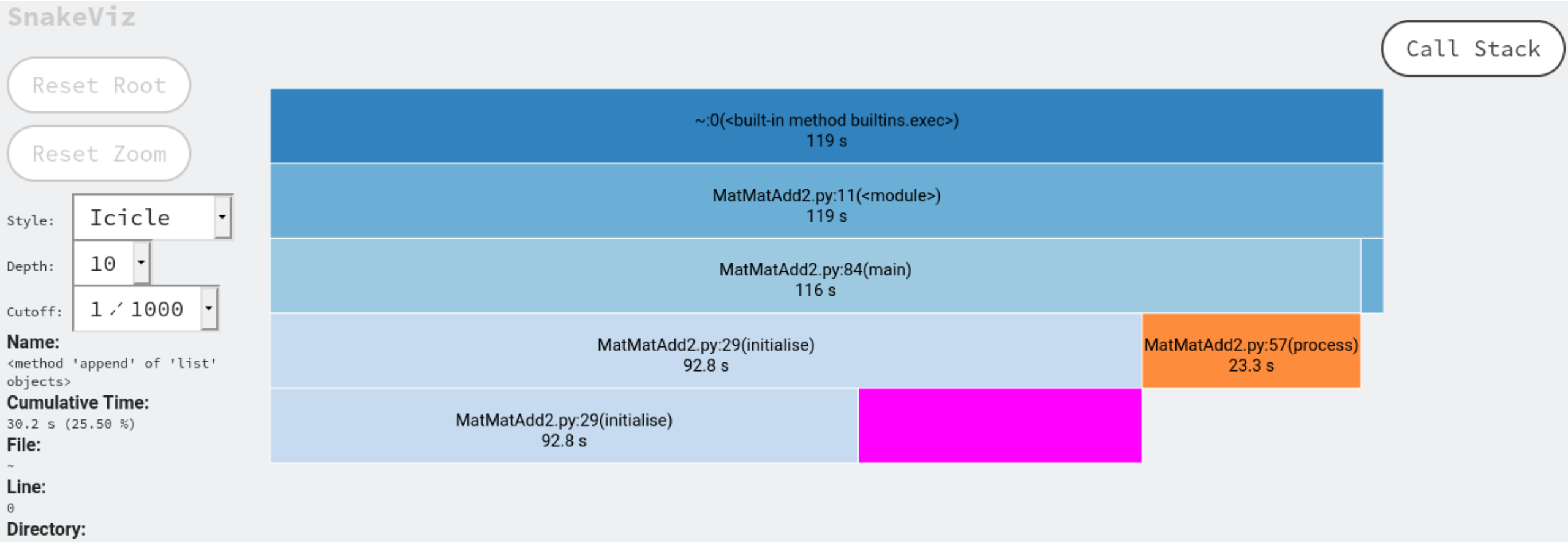
Profiling Code (`cProfile` sample)

```
805355529 function calls in 118.819 seconds

   Ordered by: cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000  118.819  118.819 {built-in method builtins.exec}
        1    2.420    2.420  118.819  118.819 MatMatAdd2.py:11(<module>)
        1    0.000    0.000  116.398  116.398 MatMatAdd2.py:84(main)
        1   62.740   62.740   93.062   93.062 MatMatAdd2.py:29(initialise)
805355520   30.322    0.000   30.322    0.000 {method 'append' of 'list' objects}
        1   23.337   23.337   23.337   23.337 MatMatAdd2.py:57(process)
        1    0.000    0.000    0.000    0.000 MatMatAdd2.py:15(setup)
        1    0.000    0.000    0.000    0.000 {built-in method builtins.print}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
        1    0.000    0.000    0.000    0.000 {built-in method builtins.len}
```

Profiling Code (SnakeViz sample)



Profiling Code (SnakeViz sample)

Search:

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	62.61	62.61	92.83	92.83	MatMatAdd2.py:29(initialise)
805355520	30.22	3.753e-08	30.22	3.753e-08	~:0(<method 'append' of 'list' objects>)
1	23.29	23.29	23.29	23.29	MatMatAdd2.py:57(process)
1	2.418	2.418	118.5	118.5	MatMatAdd2.py:11(<module>)
1	9.224e-06	9.224e-06	9.224e-06	9.224e-06	~:0(<built-in method builtins.print>)
1	5.393e-06	5.393e-06	116.1	116.1	MatMatAdd2.py:84(main)
1	3.409e-06	3.409e-06	1.294e-05	1.294e-05	MatMatAdd2.py:15(setup)
1	2.357e-06	2.357e-06	118.5	118.5	~:0(<built-in method builtins.exec>)
1	5.72e-07	5.72e-07	5.72e-07	5.72e-07	~:0(<method 'disable' of '_lsprof.Profiler' objects>)
1	3.07e-07	3.07e-07	3.07e-07	3.07e-07	~:0(<built-in method builtins.len>)

Showing 1 to 10 of 10 entries

Line profiling

- **Using the `line_profiler` package:**

```
@profile
def MyFunction ():
```

1. Add an annotation (`@profile` decorator) to your functions

```
kernprof -l -v MyPythonCode.py
```

2. Instrument your code for line profiling

```
python -m line_profiler MyPythonCode.py.lprof
```

3. Pass the instrumented code in the `MyPythonCode.py.lprof` script file to the `line_profiler` package

Profiling Code (`line_profiler` sample)

```
Timer unit: 1e-06 s

Total time: 494.288 s
File: MatMatAdd3.py
Function: initialise at line 29

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    29                                           #===( Initialisation )=====================================================
    30
    31                                           @profile
    32                                           def initialise ( matA, matB, matY, Rows, Cols ):
    33
    34     16385       4557.0      0.3      0.0    # Initialize matrix A
    35     16384    1079428.0     65.9      0.2
    36 268451840   75533955.0      0.3     15.3    for r in range ( Rows ):      # A for loop for row entries
    37 268435456   83015741.0      0.3     16.8      v = []
    38     16384       5588.0      0.3      0.0      for c in range ( Cols ):  # A for loop for column entries
    39                                                 v.append ( 1.0 )
    40                                               matA.append ( v )
    41     16385       4713.0      0.3      0.0
    42     16384    1270204.0     77.5      0.3    # Initialize matrix B
    43 268451840   78478761.0      0.3     15.9    for r in range ( Rows ):      # A for loop for row entries
    44 268435456   86529825.0      0.3     17.5      v = []
    45     16384       5662.0      0.3      0.0      for c in range ( Cols ):  # A for loop for column entries
    46                                                 v.append ( 2.0 )
    47                                               matB.append ( v )
    48     16385       5157.0      0.3      0.0
    49     16384    1270371.0     77.5      0.3    # Initialize matrix Y
    50 268451840   79801429.0      0.3     16.1    for r in range ( Rows ):      # A for loop for row entries
    51 268435456   87277163.0      0.3     17.7      v = []
    52     16384       5913.0      0.4      0.0      for c in range ( Cols ):  # A for loop for column entries
    53                                                 v.append ( 0.0 )
    54         1          1.0      1.0      0.0      matY.append ( v )
    55
    56                                             return ( matA, matB, matY )
```

Memory consumption

- **Using the `memory_profiler` package:**

```
@profile
def MyFunction ():
```

1. Add an annotation (`@profile` decorator) to your functions

```
python -m memory_profiler MyPythonCode.py
```

2. Pass the `MyPythonCode.py` script file to the `memory_profiler` package

Profiling Code (`memory_profiler` sample)

```
Filename: MatMatAdd5.py

Line #    Mem usage    Increment  Occurences   Line Contents
=============================================================
    28   70.195 MiB   70.195 MiB           1   @profile
    29                                         def initialise ( matA, matB, matY, Rows, Cols ):
    30
    31                                           # Matrix shape is NxN ( N, N )
    32 2118.391 MiB 2048.195 MiB           1     matA = np.full ( ( Rows, Cols ), 1.0, dtype = float )
    33 4166.324 MiB 2047.934 MiB           1     matB = np.full ( ( Rows, Cols ), 2.0, dtype = float )
    34
    35 4166.324 MiB    0.000 MiB           1     return ( matA, matB, matY )


Filename: MatMatAdd5.py

Line #    Mem usage    Increment  Occurences   Line Contents
=============================================================
    39 4166.324 MiB 4166.324 MiB           1   @profile
    40                                         def process ( matA, matB, matY ):
    41
    42 6214.254 MiB 2047.930 MiB           1     matY = matA + matB
    43 6214.254 MiB    0.000 MiB           1     return ( matY )
```

Pre-requisites

- `pip install snakeviz`
- `pip install line_profiler`
- `pip install memory_profiler`