AOS - Special Project

Name: Rajveer Bachkaniwala
Email: rbachkaniwala3@gatech.edu

Contents

1. Problem statement
2. Introduction
3. Code
4. Result

Problem statement:

The Linux perf profiler is a powerful tool that provides users with a range of performance profiling capabilities. One of the strengths of perf is its vibrant ecosystem of tools that aid in data analysis. However, when using perf with Python applications, there is a major issue. Perf only captures information about native symbols, such as functions and procedures written in C. As a result, any Python functions included in the code will not be included in the output of perf. It can be observed that python level function information is lost.

This problem has been solved starting python 3.12. The alpha release for python 3.12 is scheduled to be in October 2023. The current state of libraries will not be updated until later on. An issue with Python 3.12 is that it removes the common build system that libraries rely on currently to build python applications, this further delays the benefits that can be harvested from Python 3.12's perf feature.

To bypass this and enable current libraries to support this, this project proposes a patch for Python 3.11 that will enable Python 3.12's perf support.

We demonstrate the patched Python 3.11's by getting the same profile results using perf as Python 3.12.

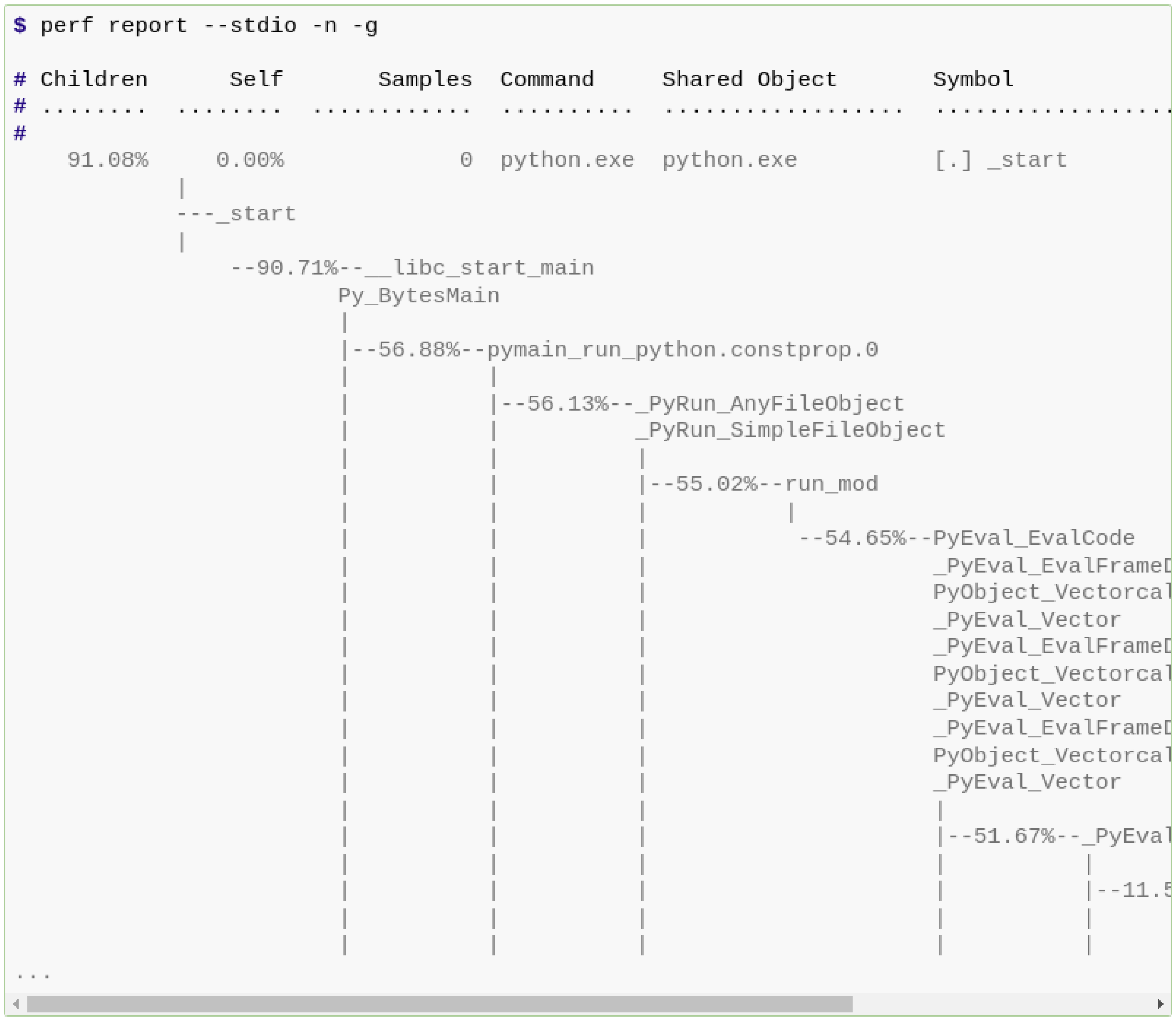
An Example, from python docs, to demonstrate the problem with some python code in a file called "*test.py*":

```python
def foo(n):
    result = 0
    for _ in range(n):
        result += 1
    return result

def bar(n):
    foo(n)

def baz(n):
    bar(n)

if __name__ == "__main__":
    baz(1000000)
```

We run below command to collect traces:

```
$ perf record -F 9999 -g -o perf.data python my_script.py
```

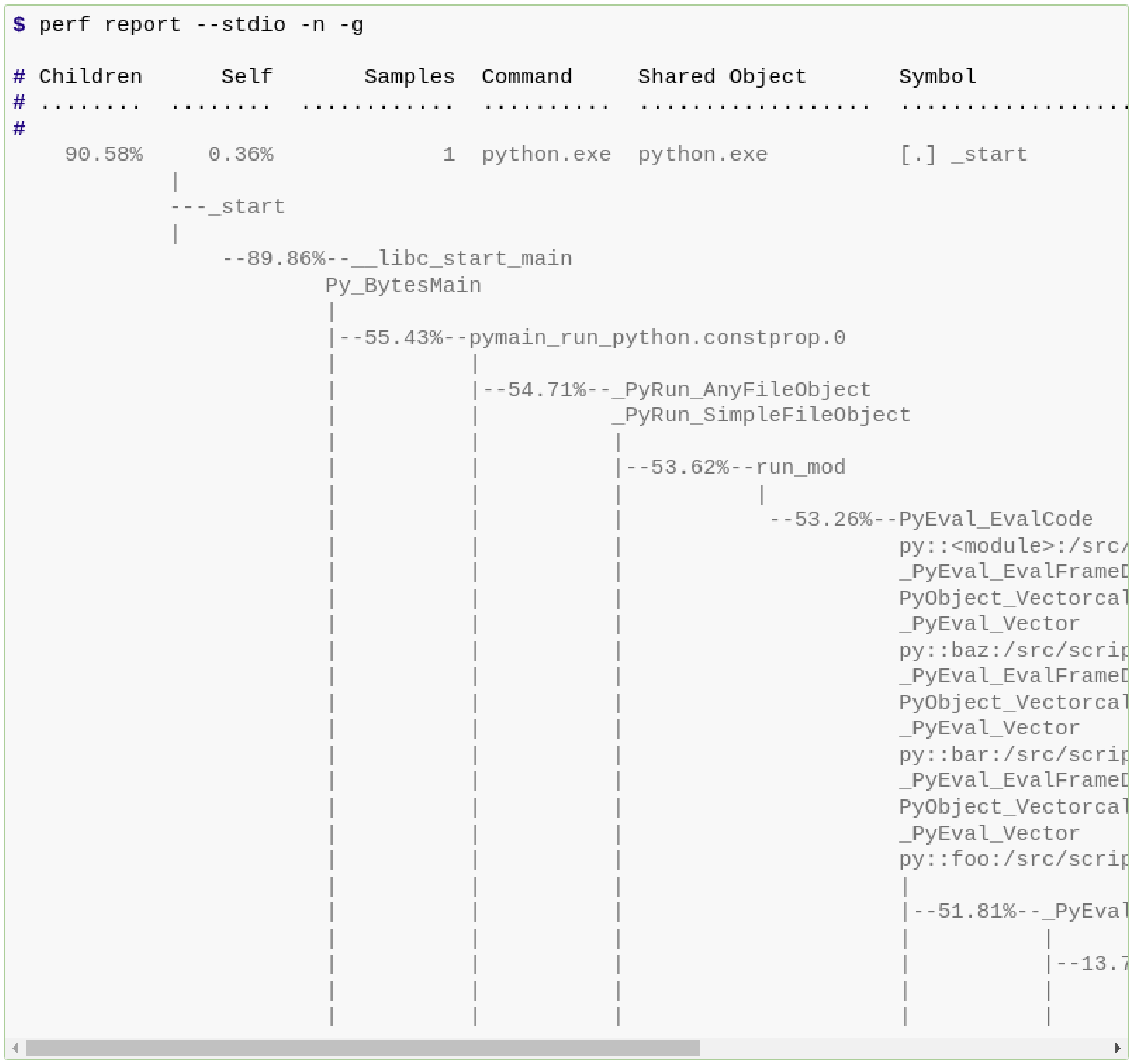
Below is the trace for above command:
Notice that python level function information is missing.



```
$ perf report --stdio -n -g

# Children      Self       Samples  Command     Shared Object       Symbol
# ........  ........  ............  ..........  ..................  ..................
#
    91.08%     0.00%                 0  python.exe  python.exe          [.] _start
            |
            ---_start
            |
                --90.71%--__libc_start_main
                        Py_BytesMain
                        |
                        |--56.88%--pymain_run_python.constprop.0
                        |          |
                        |          |--56.13%--_PyRun_AnyFileObject
                        |          |          _PyRun_SimpleFileObject
                        |          |          |
                        |          |          |--55.02%--run_mod
                        |          |          |          |
                        |          |          |           --54.65%--PyEval_EvalCode
                        |          |          |                     _PyEval_EvalFrameD
                        |          |          |                     PyObject_Vectorcal
                        |          |          |                     _PyEval_Vector
                        |          |          |                     _PyEval_EvalFrameD
                        |          |          |                     PyObject_Vectorcal
                        |          |          |                     _PyEval_Vector
                        |          |          |                     _PyEval_EvalFrameD
                        |          |          |                     PyObject_Vectorcal
                        |          |          |                     _PyEval_Vector
                        |          |          |                     |
                        |          |          |                     |--51.67%--_PyEval
                        |          |          |                     |          |
                        |          |          |                     |          |--11.5
                        |          |          |                     |          |
                        |          |          |                     |          |
...
```

Check below for Python 3.12's profile and what is expected from Python 3.11:

Below is the profile post enabling perf support:



```
$ perf report --stdio -n -g

# Children      Self       Samples  Command     Shared Object       Symbol
# ........  ........  ............  ..........  ..................  ...................
#
    90.58%     0.36%             1  python.exe  python.exe          [.] _start
            |
            ---_start
            |
                --89.86%--__libc_start_main
                        Py_BytesMain
                        |
                        |--55.43%--pymain_run_python.constprop.0
                        |          |
                        |          |--54.71%--_PyRun_AnyFileObject
                        |          |          _PyRun_SimpleFileObject
                        |          |          |
                        |          |          |--53.62%--run_mod
                        |          |          |          |
                        |          |          |           --53.26%--PyEval_EvalCode
                        |          |          |                     py::<module>:/src/
                        |          |          |                     _PyEval_EvalFrameD
                        |          |          |                     PyObject_Vectorcal
                        |          |          |                     _PyEval_Vector
                        |          |          |                     py::baz:/src/scrip
                        |          |          |                     _PyEval_EvalFrameD
                        |          |          |                     PyObject_Vectorcal
                        |          |          |                     _PyEval_Vector
                        |          |          |                     py::bar:/src/scrip
                        |          |          |                     _PyEval_EvalFrameD
                        |          |          |                     PyObject_Vectorcal
                        |          |          |                     _PyEval_Vector
                        |          |          |                     py::foo:/src/scrip
                        |          |          |                     |
                        |          |          |                     |--51.81%--_PyEval
                        |          |          |                     |          |
                        |          |          |                     |          |--13.7
                        |          |          |                     |          |
                        |          |          |                     |          |
```

The aim of the project is to create the same profile, but with Python 3.11 instead and this project demonstrates the patch by generating flamegraphs as well.

Introduction:

For many users, this limitation of perf can be a major stumbling block. However, starting from Python version 3.12, the interpreter can run in a special mode that allows Python functions to be included in the output of the perf profiler. This is a significant step forward for developers working with Python, as it enables them to capture performance data on their Python functions in the same way as they would for C functions.

So, how does this mode work, and what are the benefits of using it? In this article, we'll explore these questions in more detail and provide a step-by-step guide to enabling the mode.

Firstly, let's examine the issue in more detail. When perf is run on a Linux system, it gathers performance data from a variety of sources. These sources include hardware performance counters, kernel tracing events, and profiling data from user-space applications. In order to capture this data accurately, perf needs to know which functions and procedures are being called by the application.

The way that perf identifies these functions is by examining the executable code that is being run. When the application is compiled, symbols are added to the binary code that correspond to the names of the functions and procedures. When perf runs, it reads these symbols and uses them to identify which functions are being called and how often they are being called.

However, when it comes to Python functions, things get more complicated. Unlike C functions, Python functions are not compiled into machine code. Instead, they are interpreted by the Python interpreter at runtime. This means that the names and file names of Python functions are not included in the binary code, and so perf is unable to identify them.

This is where the new Python mode comes in. When the interpreter is run in this mode, it inserts a small piece of code before the execution of every Python function. This code is compiled on the fly and enables perf to associate the code with the corresponding Python function using perf map files.

Code:

1. Run the below command to install perf on your system:

```
$ sudo apt-get install linux-tools-common linux-tools-generic linux-tools-`uname -r`
```

2. Code for Python patched 3.11 is available here as pre-release.
3. After pulling patched Python 3.11, navigate to the repo directory and run:

```
$ chmod +x debug_build.sh
```

```
$ sudo ./debug_build.sh
```

   Note: It is ok for some tests to fail that are not supported by your platform.
4. Above command will create a `debug` directory with the python executable inside it.
5. Run the below command to create a virtual environment and activate it.

```
$ ./debug/python -m venv ../py3_11_with_perf
```

```
$ cd .. && source ./py3_11_with_perf/bin/activate
```

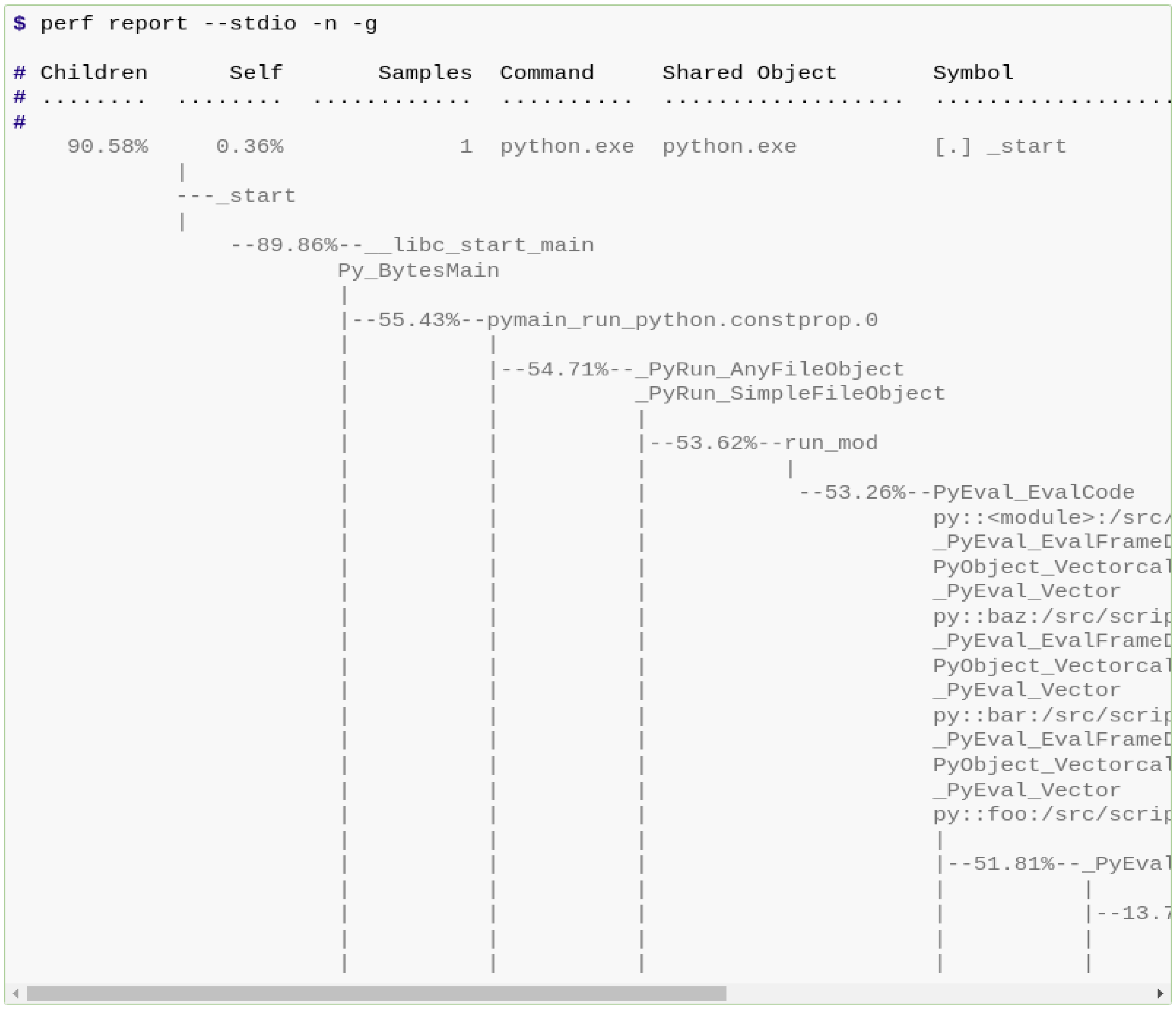
6. Run the code given above as an example with below command:

```
$ perf record -F 9999 -g -o perf.data python -X perf my_script.py
```

7. Program code and code to generate flame graphs is in this repo.

Result:

1. Below is the replication of the sample perf report present in problem statement part using Python patched 3.11 :



```
$ perf report --stdio -n -g

# Children      Self       Samples  Command     Shared Object       Symbol
# ........  ........  ............  ..........  ..................  ..................
#
    90.58%     0.36%             1  python.exe  python.exe          [.] _start
            |
            ---_start
            |
                --89.86%--__libc_start_main
                        Py_BytesMain
                        |
                        |--55.43%--pymain_run_python.constprop.0
                        |          |
                        |          |--54.71%--_PyRun_AnyFileObject
                        |          |          _PyRun_SimpleFileObject
                        |          |          |
                        |          |          |--53.62%--run_mod
                        |          |          |          |
                        |          |          |           --53.26%--PyEval_EvalCode
                        |          |          |                     py::<module>:/src/
                        |          |          |                     _PyEval_EvalFrameD
                        |          |          |                     PyObject_Vectorcal
                        |          |          |                     _PyEval_Vector
                        |          |          |                     py::baz:/src/scrip
                        |          |          |                     _PyEval_EvalFrameD
                        |          |          |                     PyObject_Vectorcal
                        |          |          |                     _PyEval_Vector
                        |          |          |                     py::bar:/src/scrip
                        |          |          |                     _PyEval_EvalFrameD
                        |          |          |                     PyObject_Vectorcal
                        |          |          |                     _PyEval_Vector
                        |          |          |                     py::foo:/src/scrip
                        |          |          |                     |
                        |          |          |                     |--51.81%--_PyEval
                        |          |          |                     |          |
                        |          |          |                     |          |--13.7
                        |          |          |                     |          |
                        |          |          |                     |          |
```

2. Code for
   Below are the links for flamegraphs (Note: Need to download and open in browser to search inside the flamegraph):
   a. `test.py` run with patched python 3.11
   b. `test.py` run with python 3.11
   c. `torch.py` run with patched python 3.11
   d. `torch.py` run with python 3.11