

NC STATE

DeepProf: Interpreting Performance Profiles with Deep Learning

Qidong Zhao (Advisor: Dr. Xu Liu)

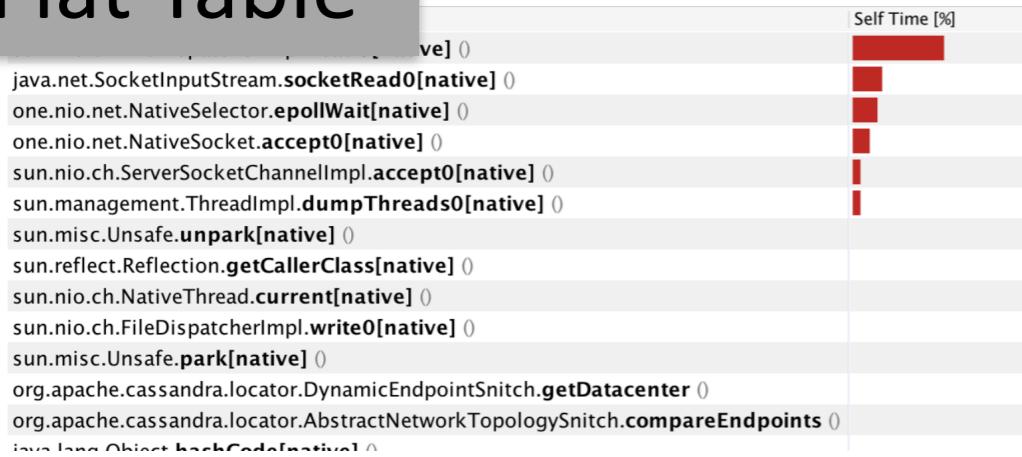
qzhao24@ncsu.edu

North Carolina State University

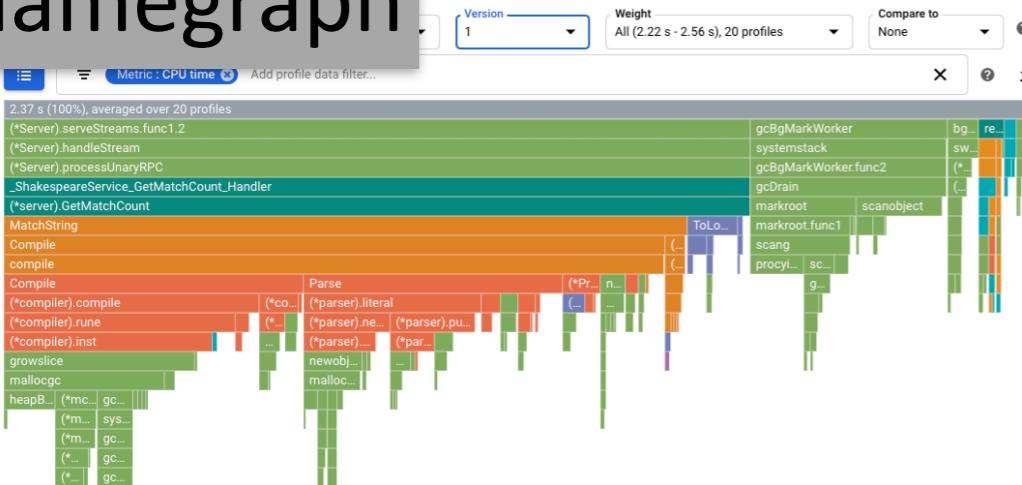
Scalable Tools Workshop
Granlibakken, Lake Tahoe, California
June 20 2023

Exploring the Visualizations of Profiles

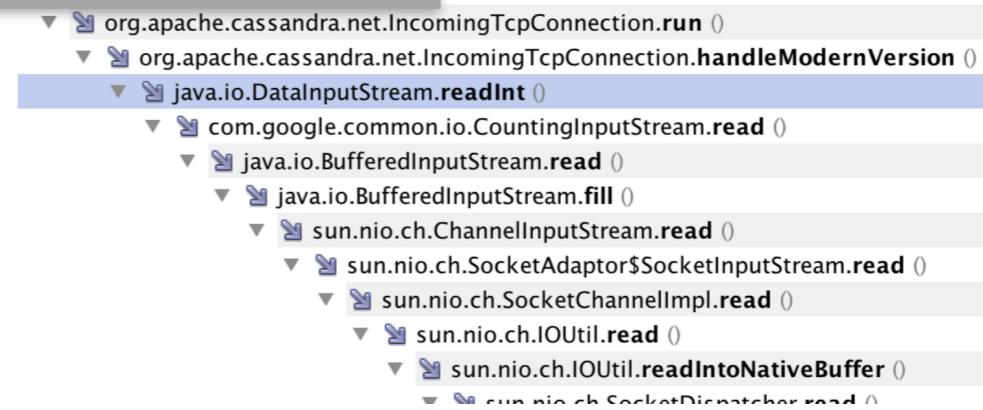
Flat Table



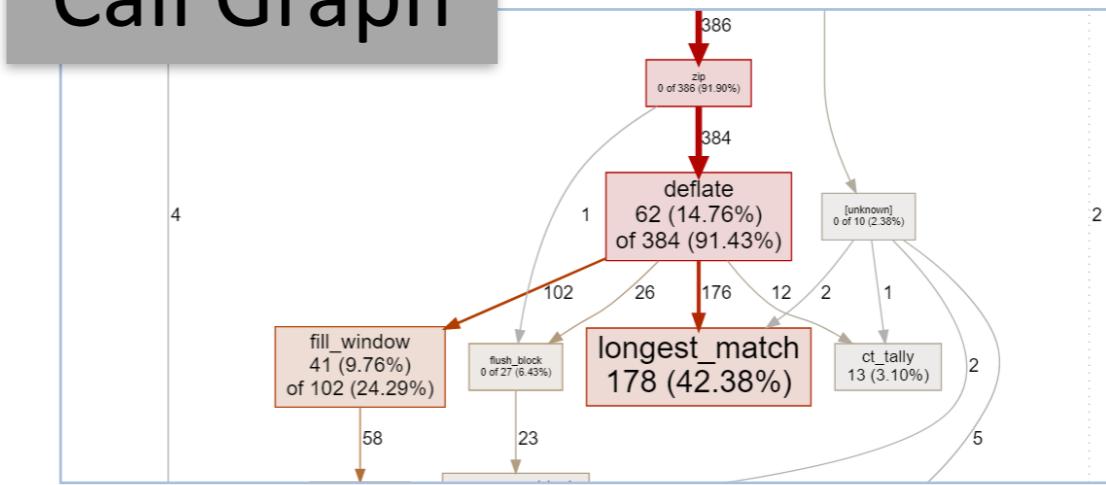
Flamegraph



Tree Table

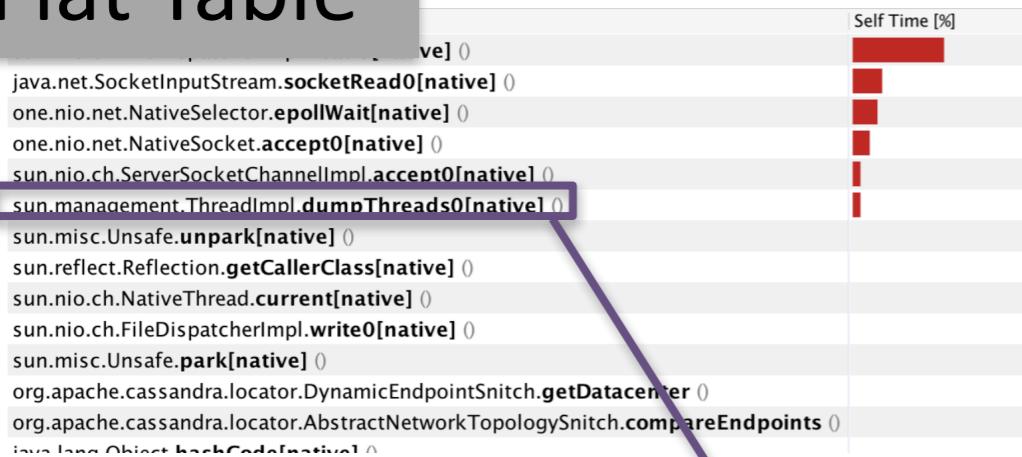


Call Graph

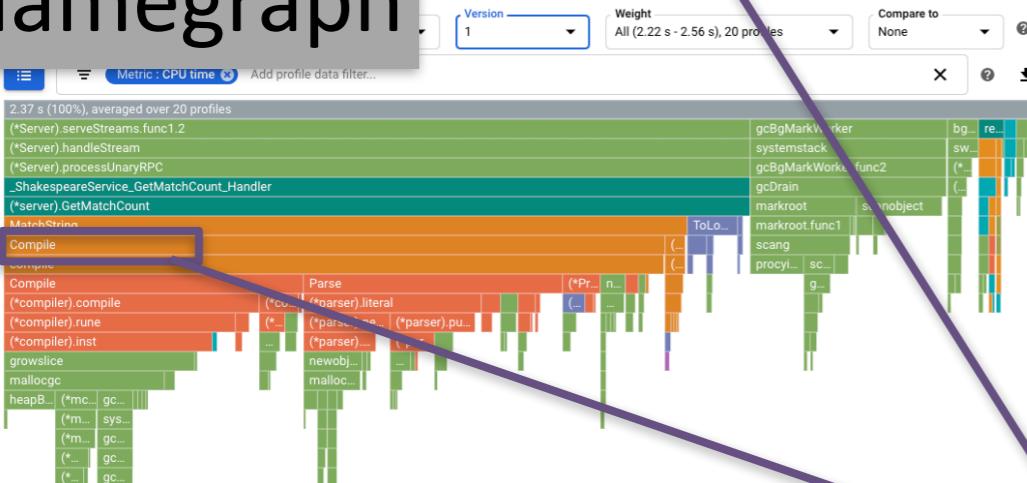


Exploring the Visualizations of Profiles

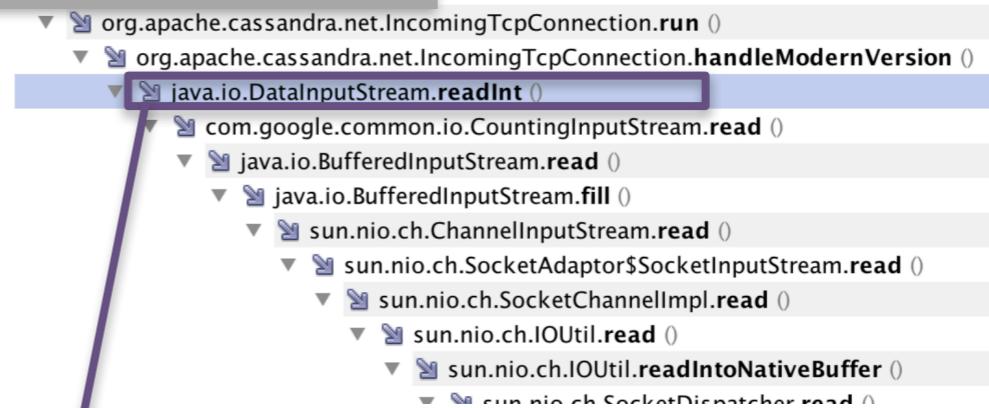
Flat Table



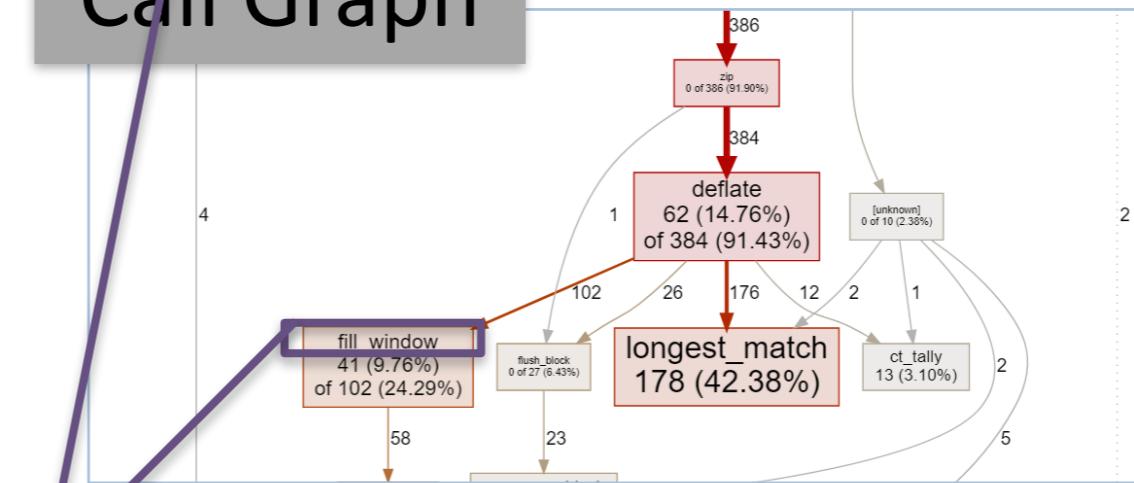
Flamegraph



Tree Table

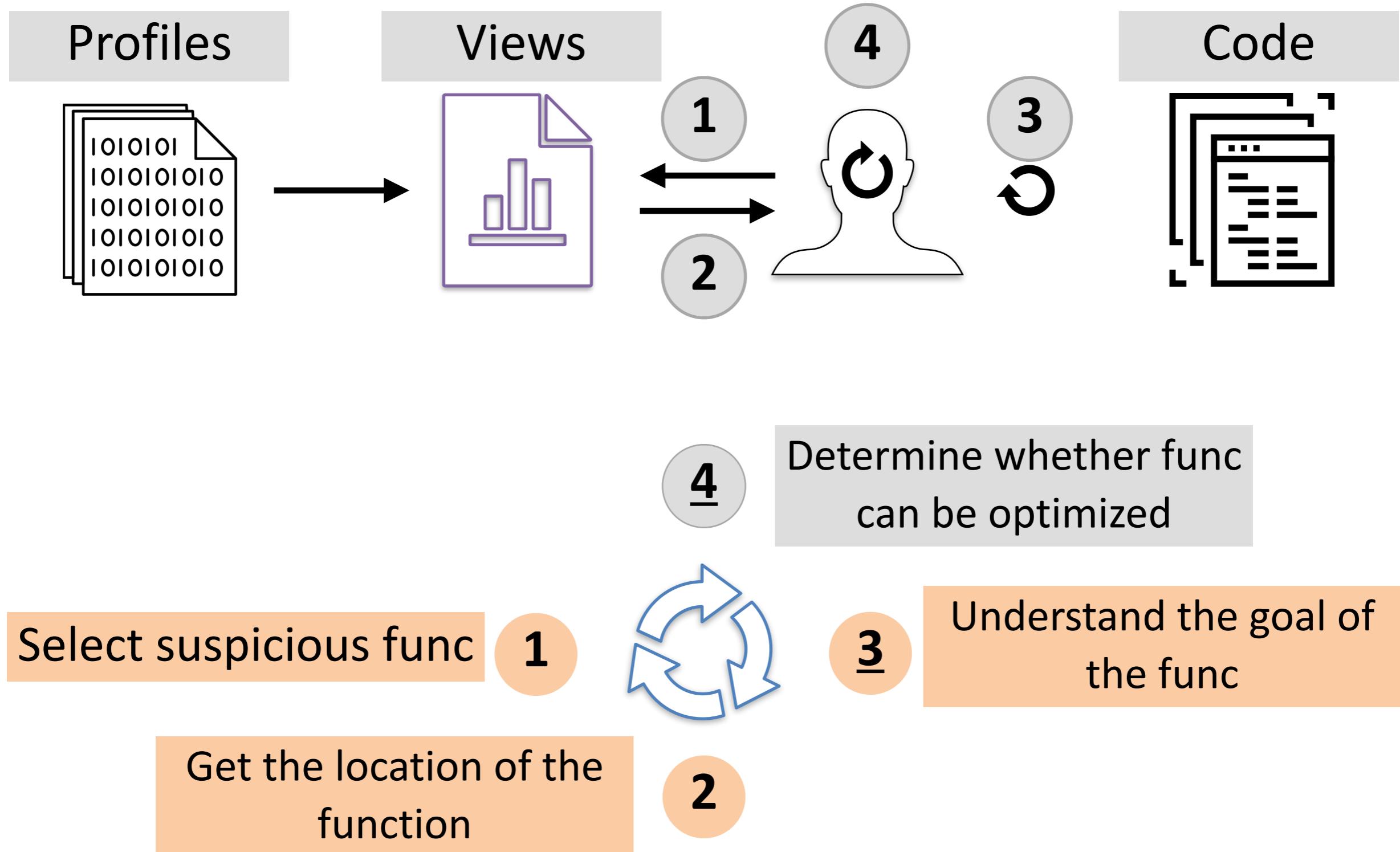


Call Graph

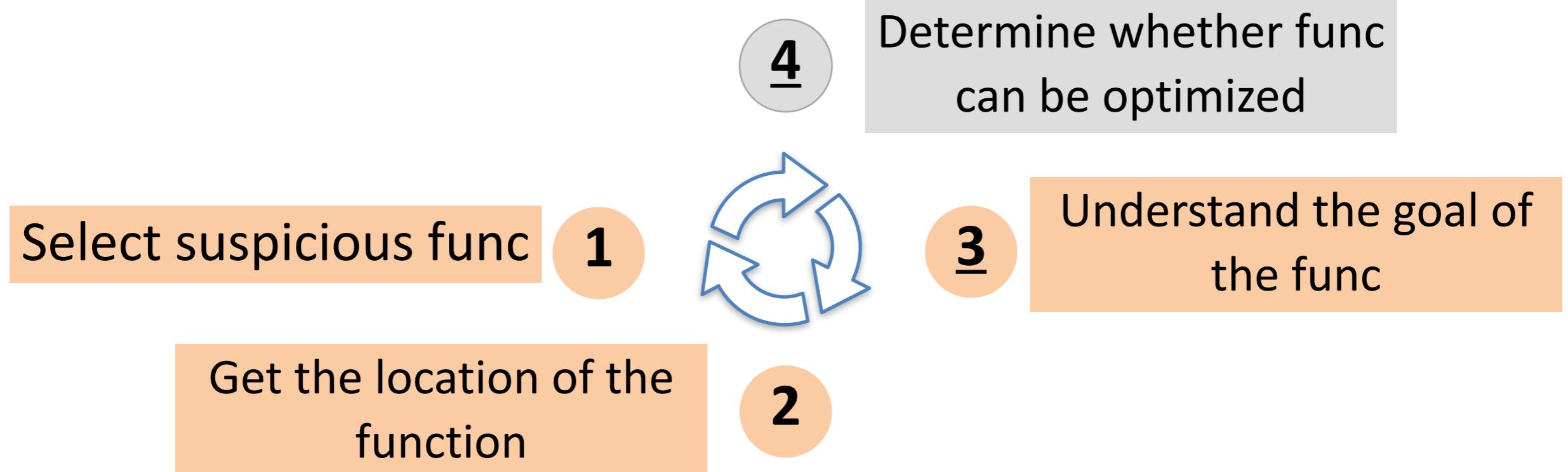


The function names and line numbers serve as indices, guiding developers directly to the relevant portion of the source code.

Exploring the Visualizations of Profiles



Challenges in Interpreting Profiles

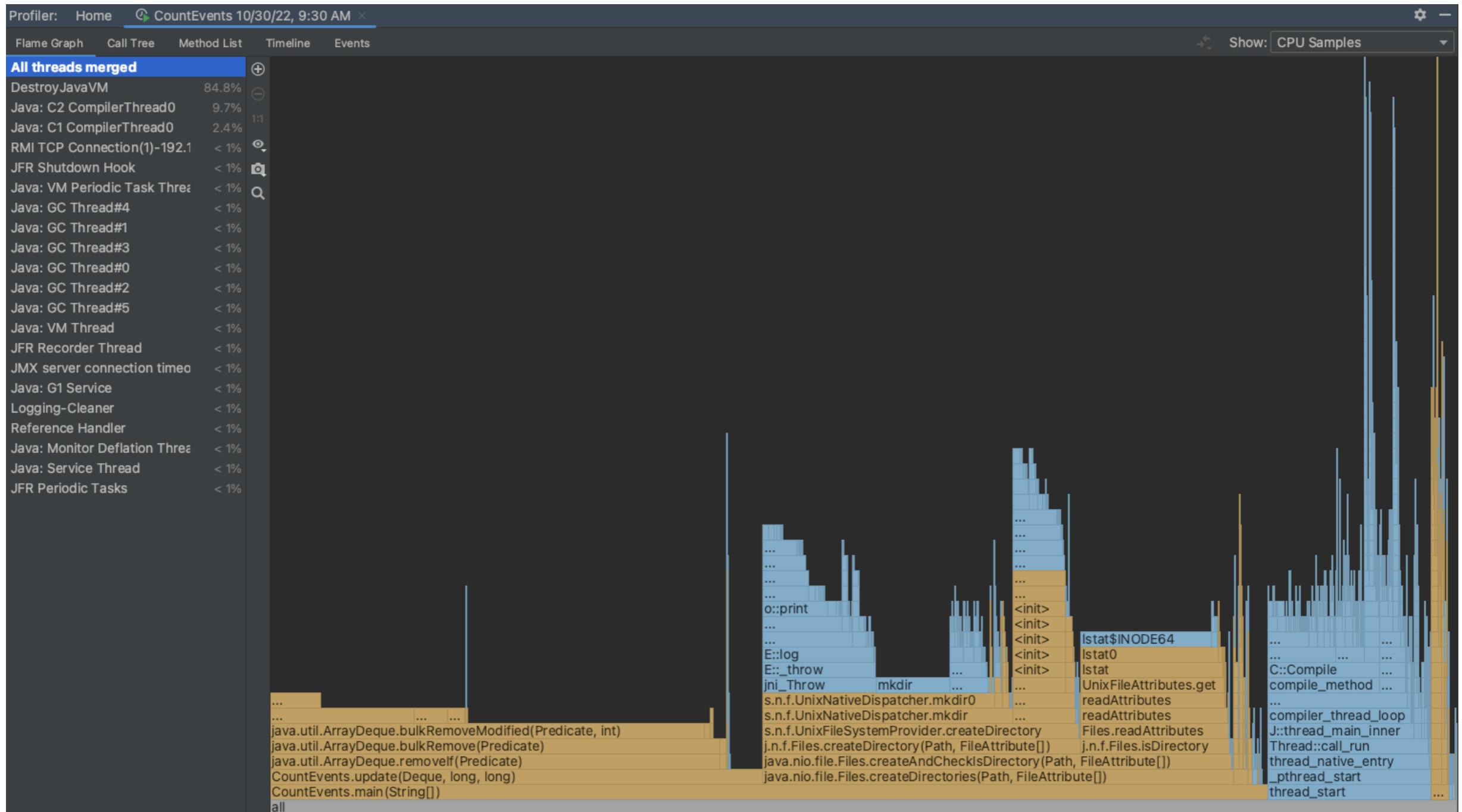


1 ✗ Not able to understand a function's goal from its name

2 ✗ Missing information for mapping to source code

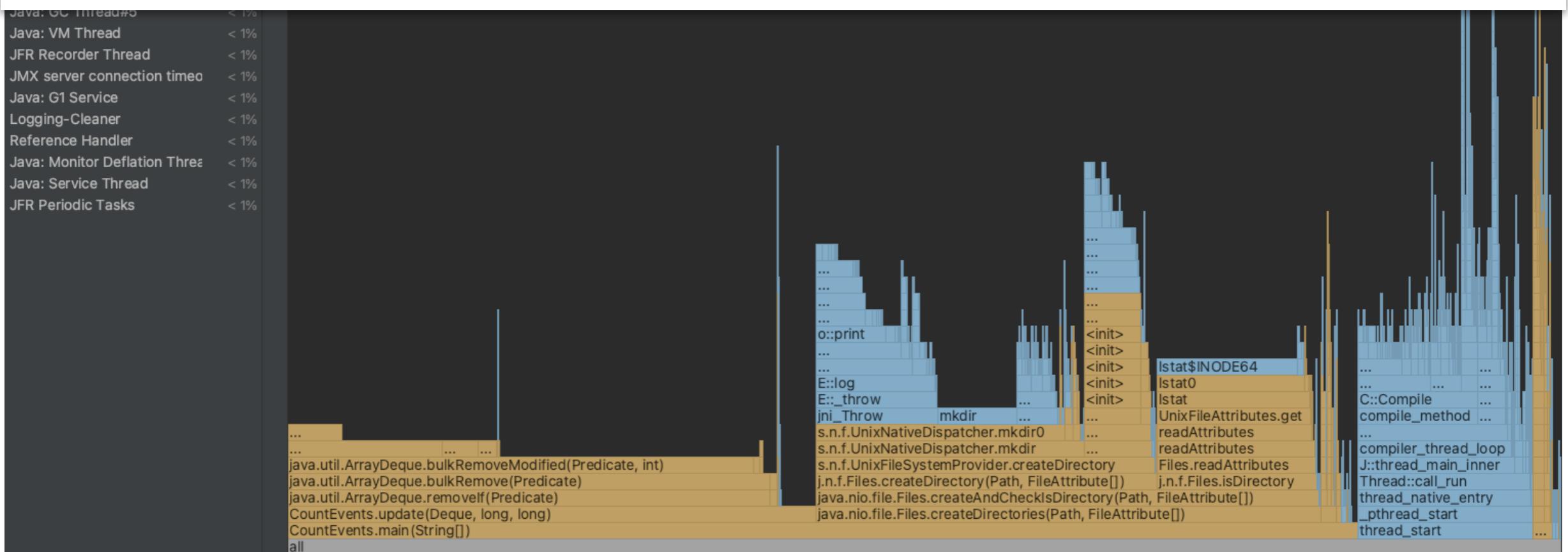
3 ✗ Not easy to understand the code of complex applications

Profile generated by Async-Profiler

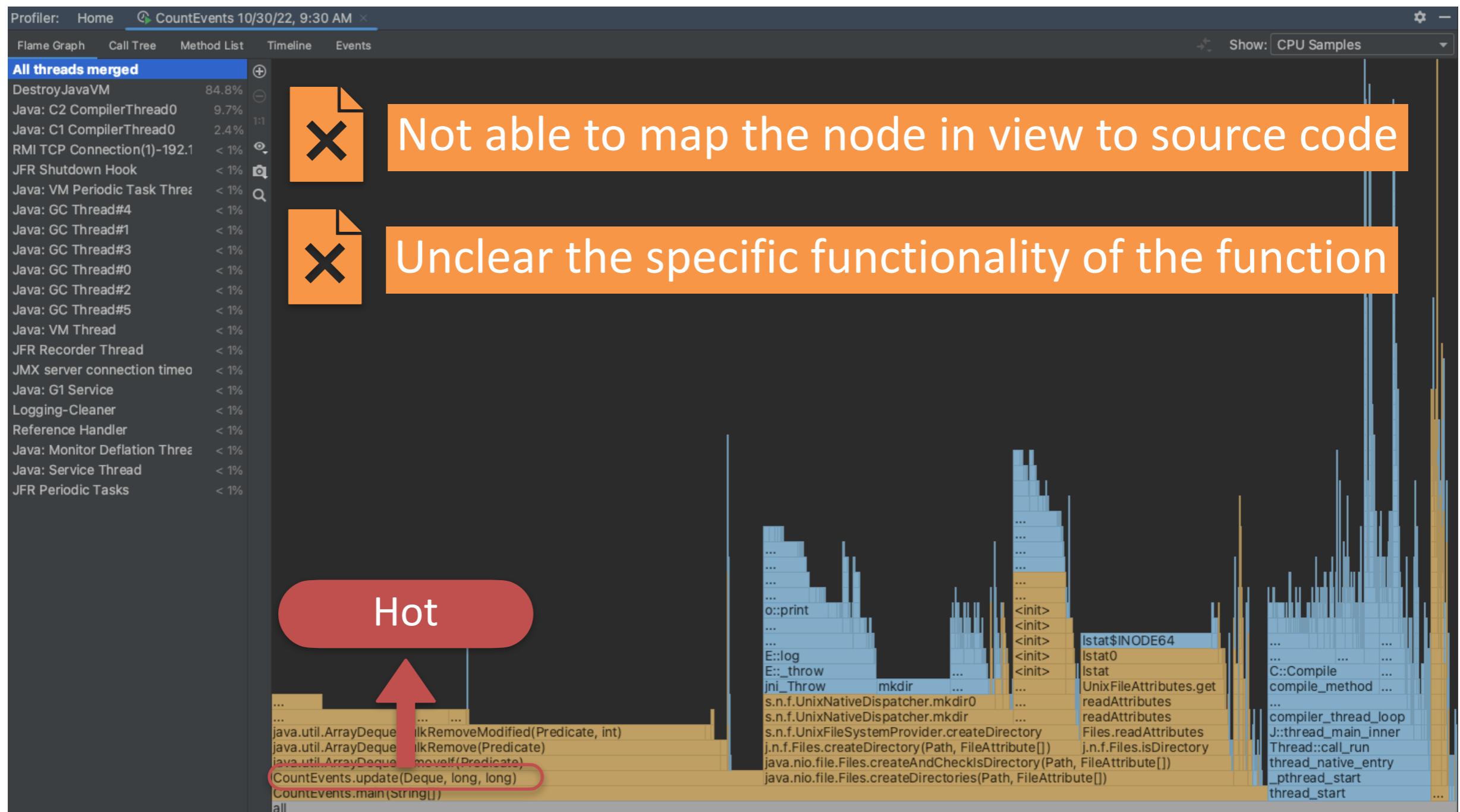


Profile generated by Async-Profiler

- Sampling-based Java profiler
- Low overhead
- Collect numerous hardware and software metrics and link them to the call stack



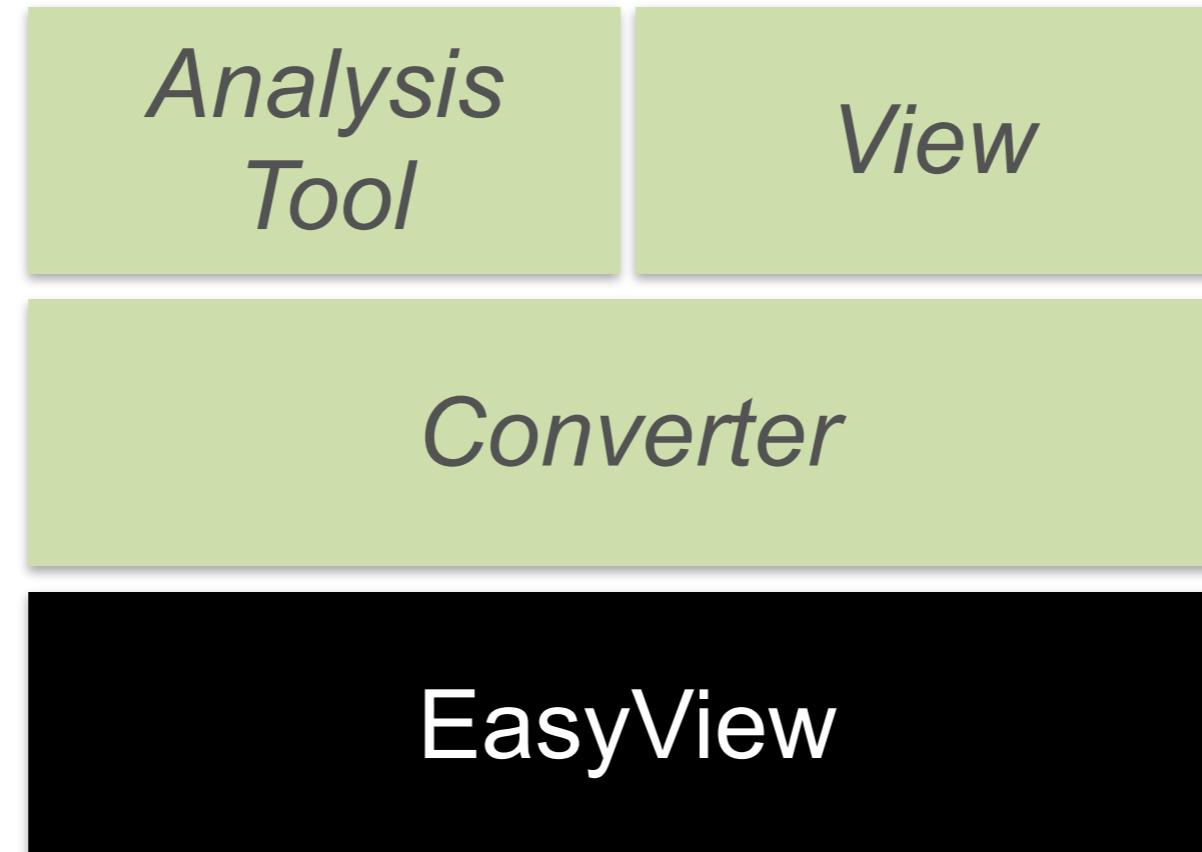
Profile generated by Async-Profiler



What is DeepProf?

Add-ons of **EasyView**

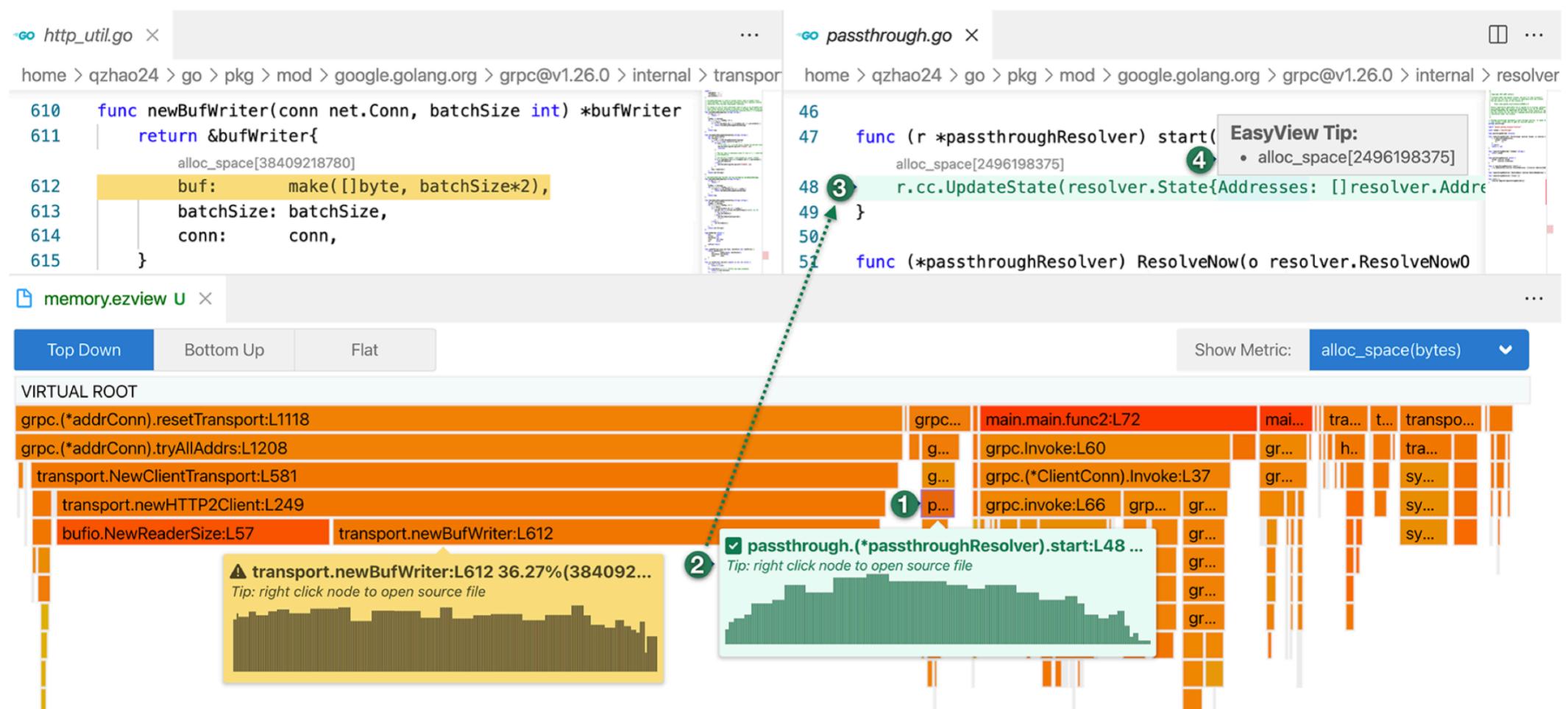
- Customized Profiles Converter
- Customized analysis tool
- Customized view



Background-EasyView

EasyView bridges the gaps for analysis and visualization
Features

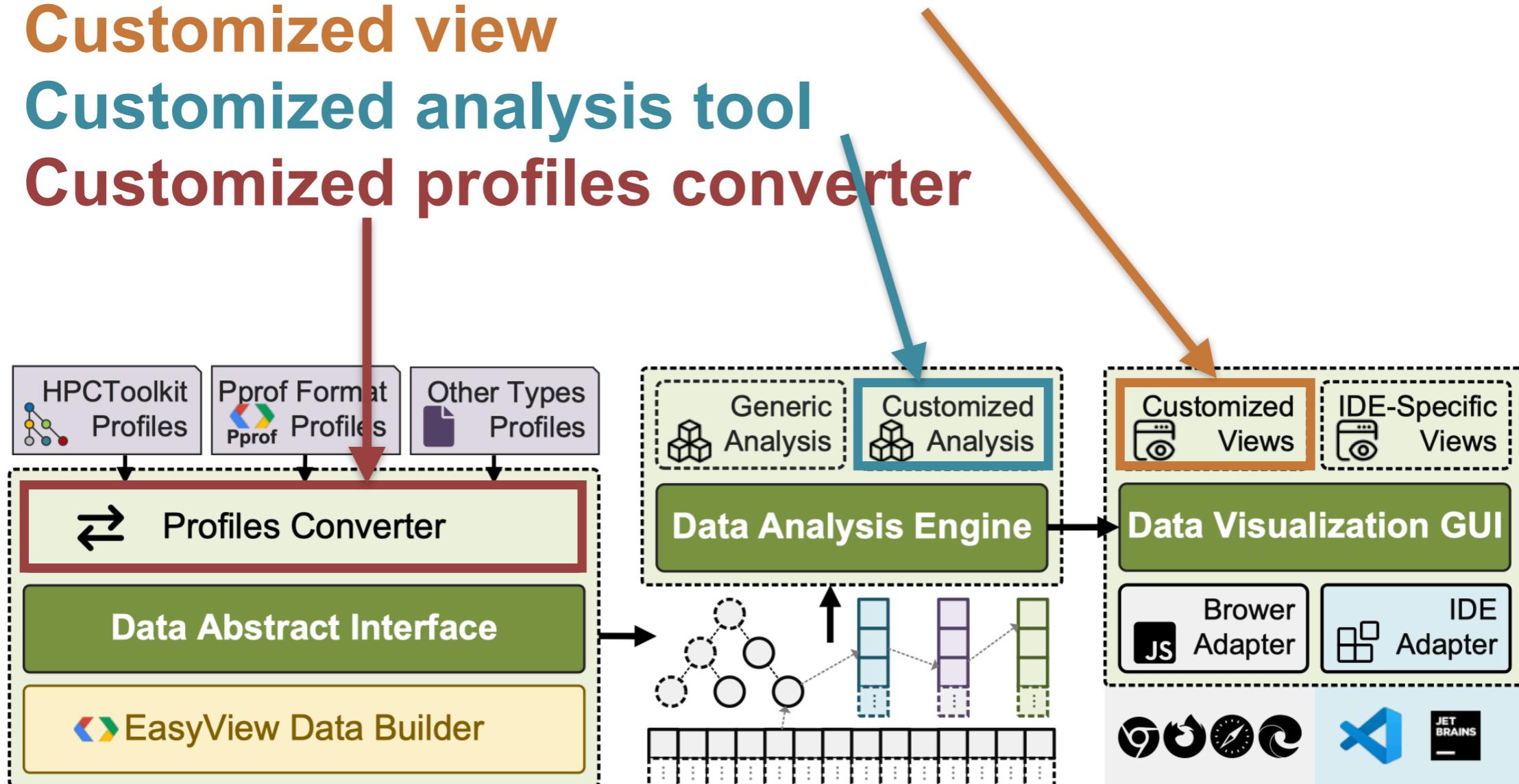
- Support many profile format
- Integrate profile analysis with development environment
- **Support advanced extension**



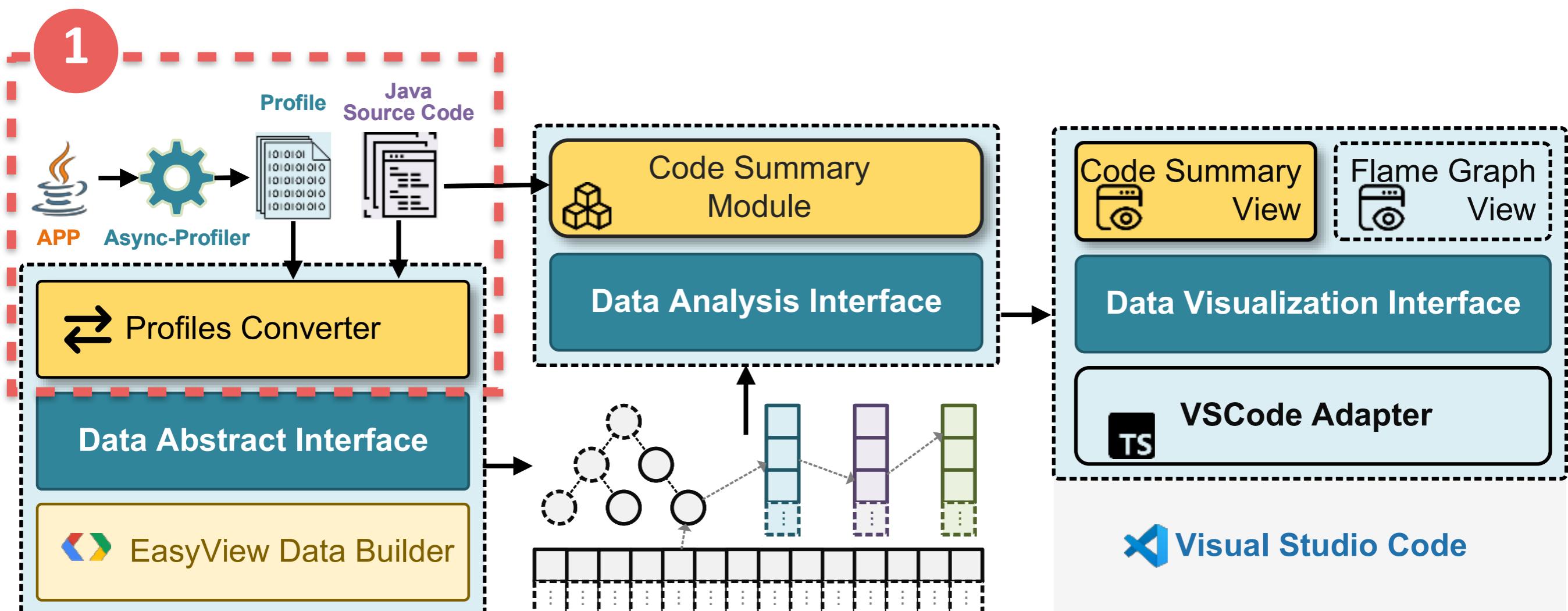
Background-EasyView

Providing interfaces that allow creating

- **Customized view**
- **Customized analysis tool**
- **Customized profiles converter**



DeepProf

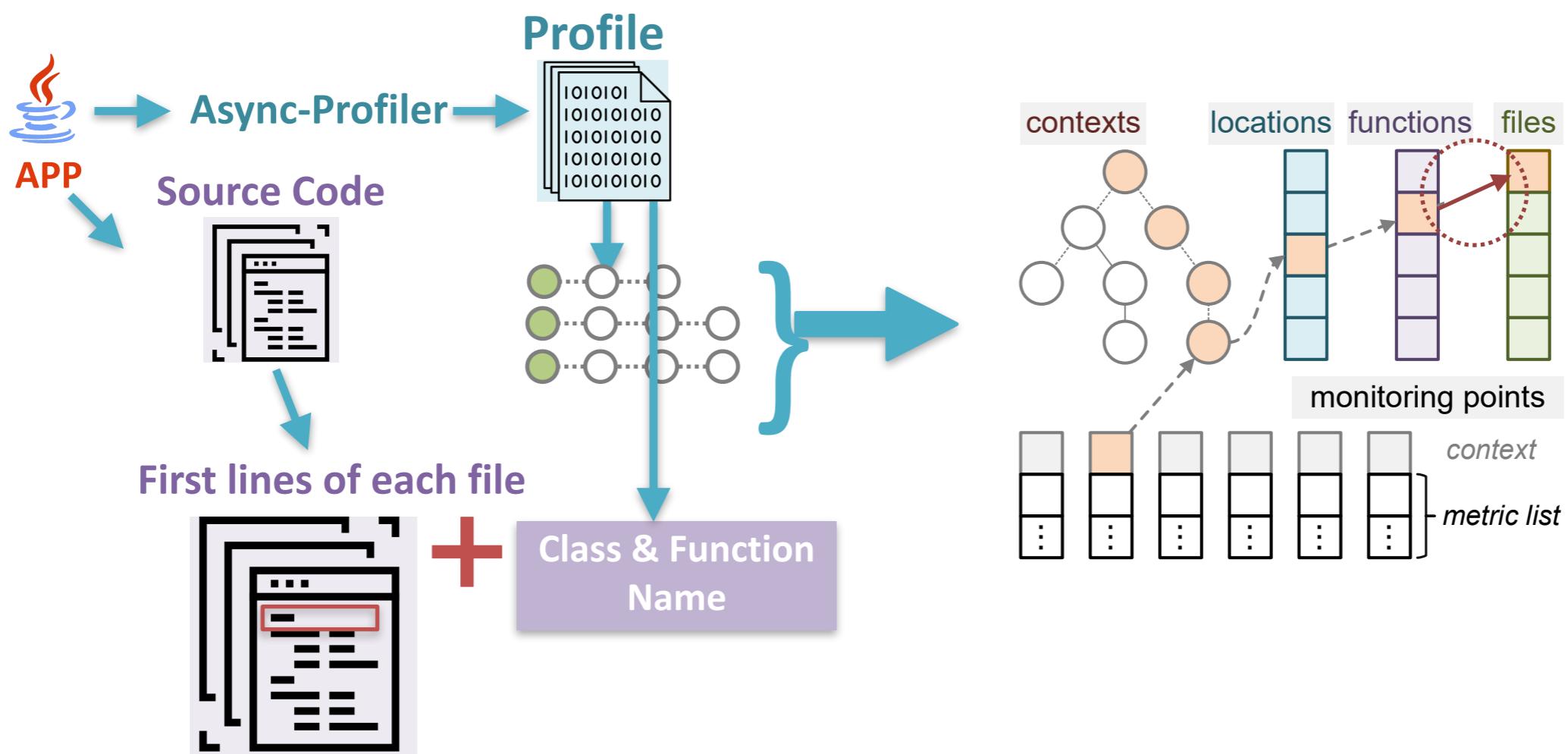


Missing information for
mapping to source file

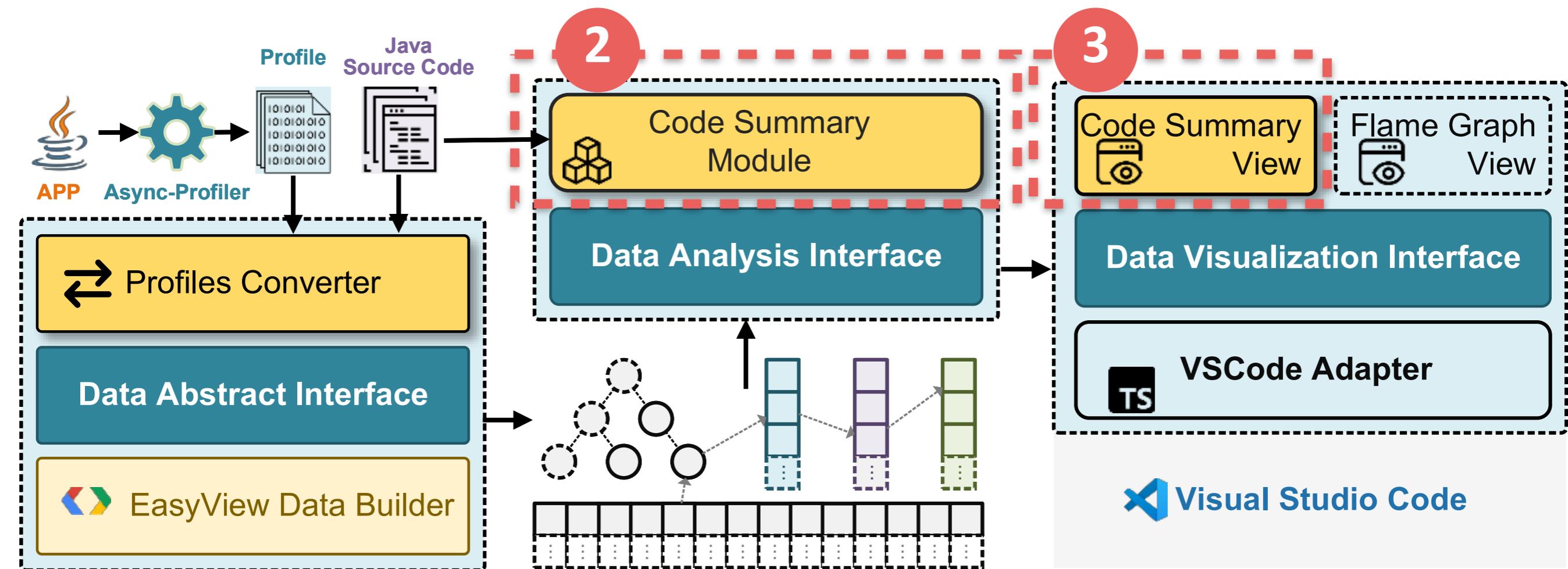
1. Automatically Enhance the Profiles

Automatically Enhance the Profiles

Mapping Profiles to Source Code



DeepProf



Miss high-level
program semantics

- 2. Code Summary Model
- 3. Call Path Summary View

Code Summary Model

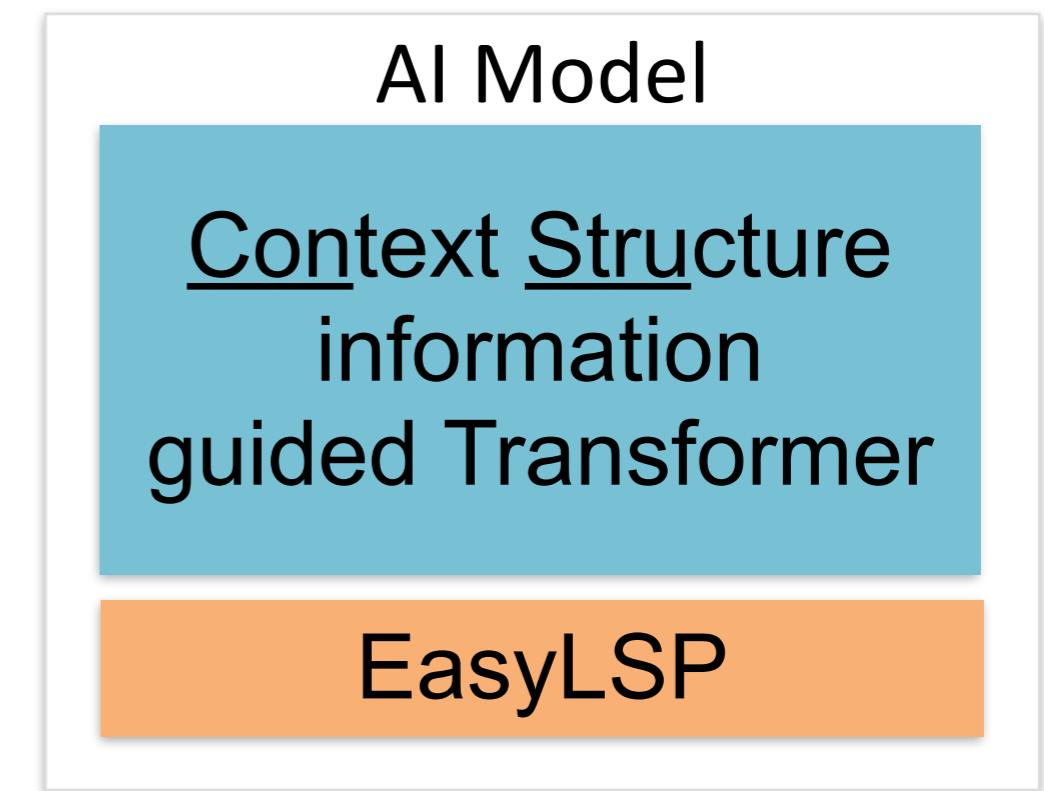
Model Architecture

The state of the art model

- ✖ *Function name sensitively*
- ✖ *Missing Context Structural information*

Our code summary model

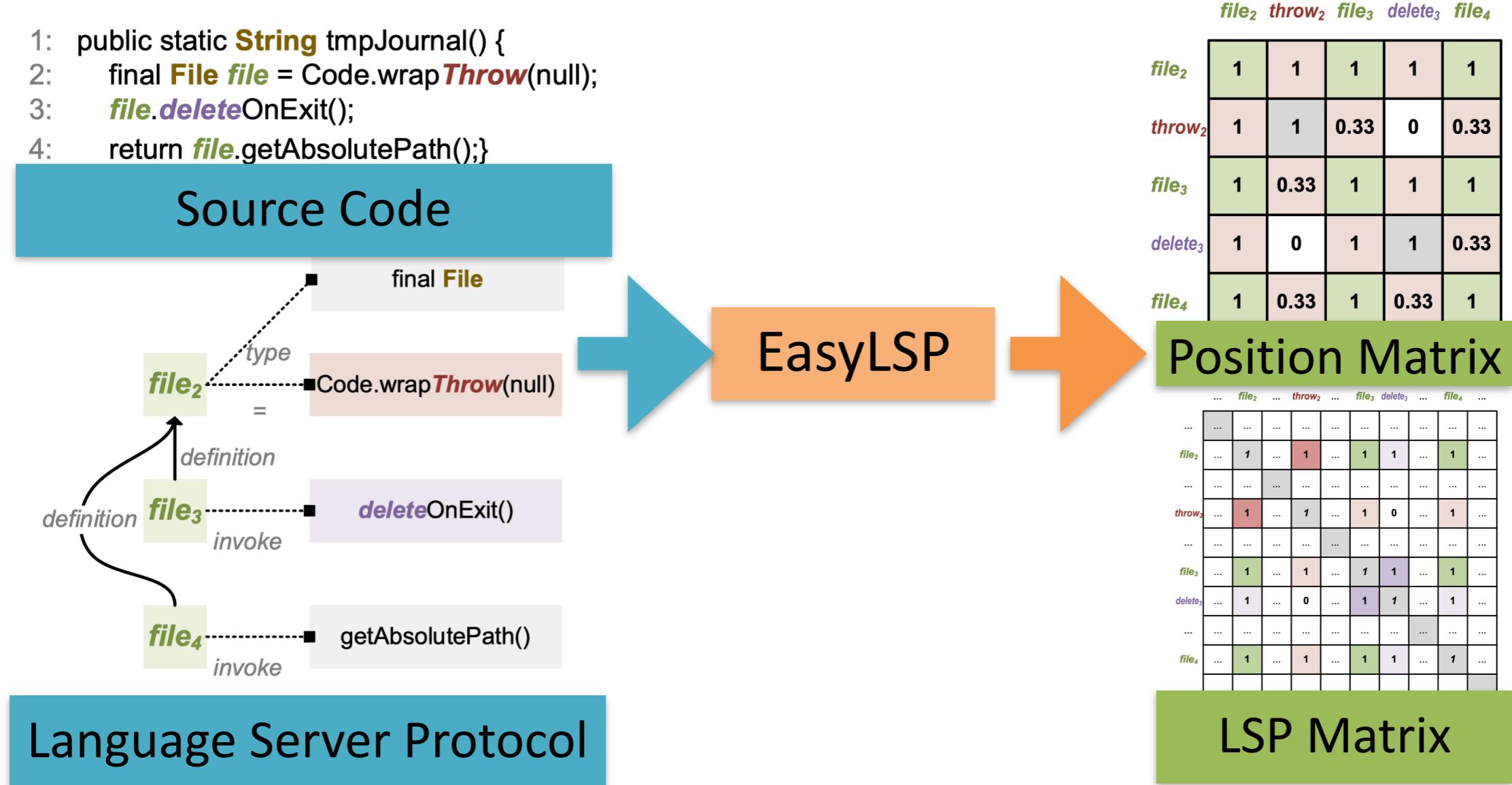
- **EasyLSP**
- **ConStruct**



Code Summary Model

EasyLSP

- extract structural information directly from the source code via language server protocol (LSP)



Context Structure information guided Transformer

ConStruct

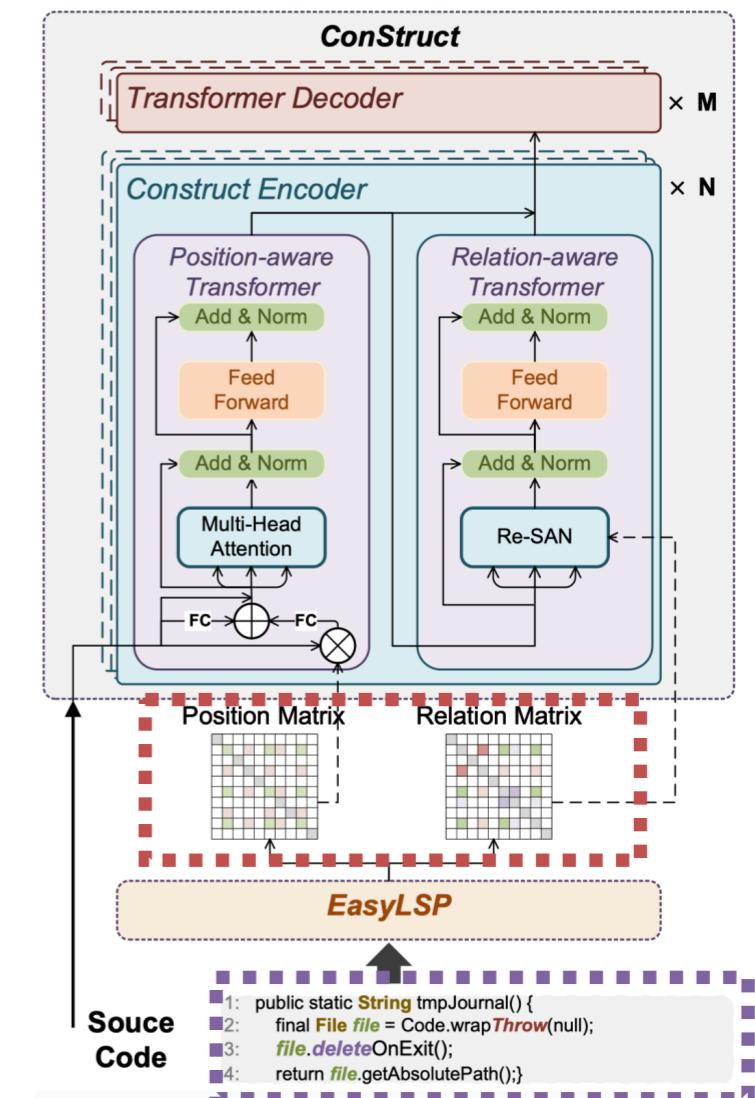
a context structure information guided transformer

(1) A novel attention mechanism

(2) A new encode

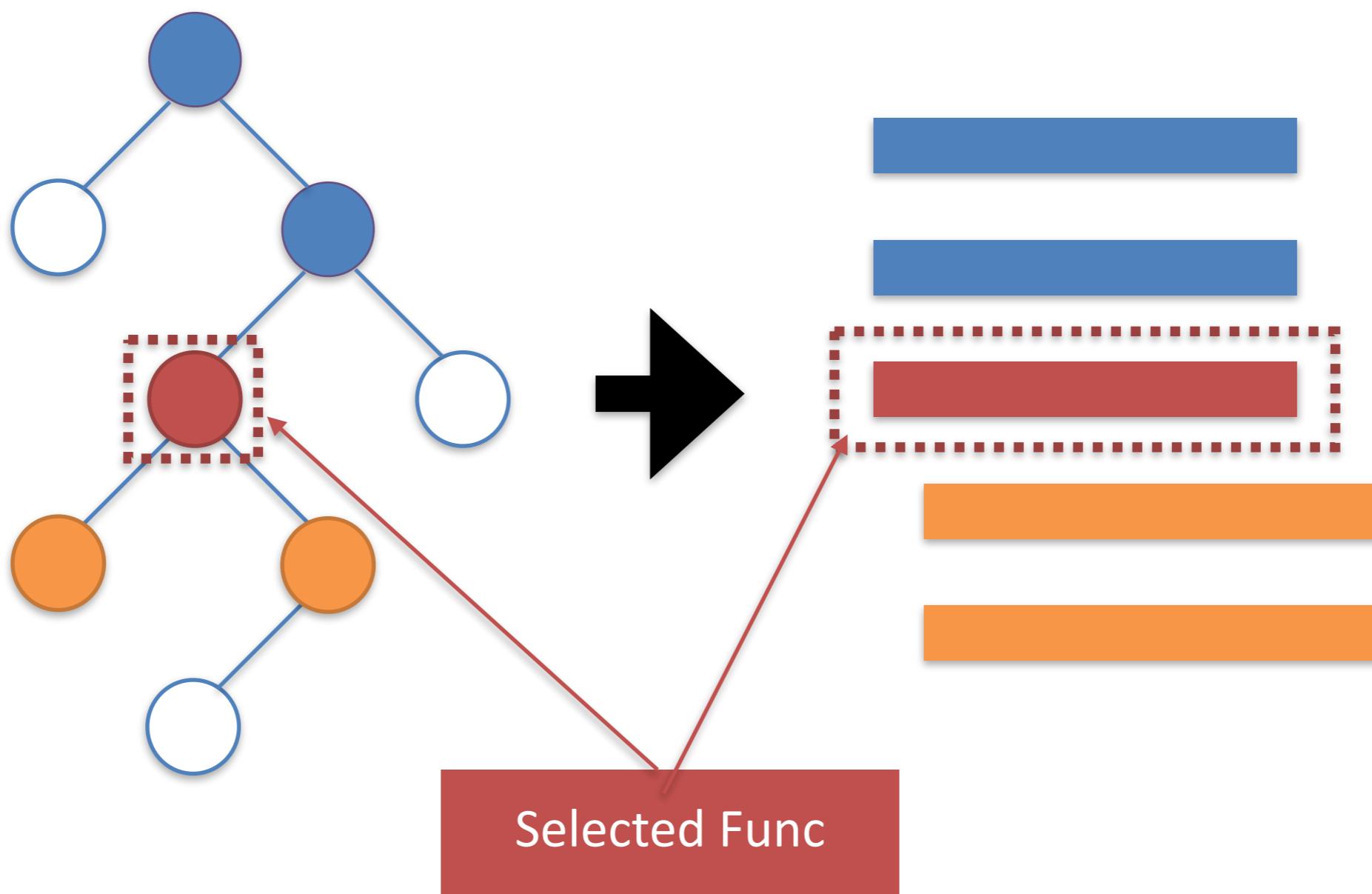
- sequential code inputs
- relation matrix
- position matrix

(3) Improve performance as evaluated by standard metrics.(BLEU+10.5%, ROUGE-L+4.8%, and METEOR+5.2%)



*BLEU/ROUGE/METEOR: metrics used to evaluate NLP model

Call Path Summary View



Case-Directly identify the possible optimization

Micro Benchmark

SortRunner.java

A class provide quickSort()

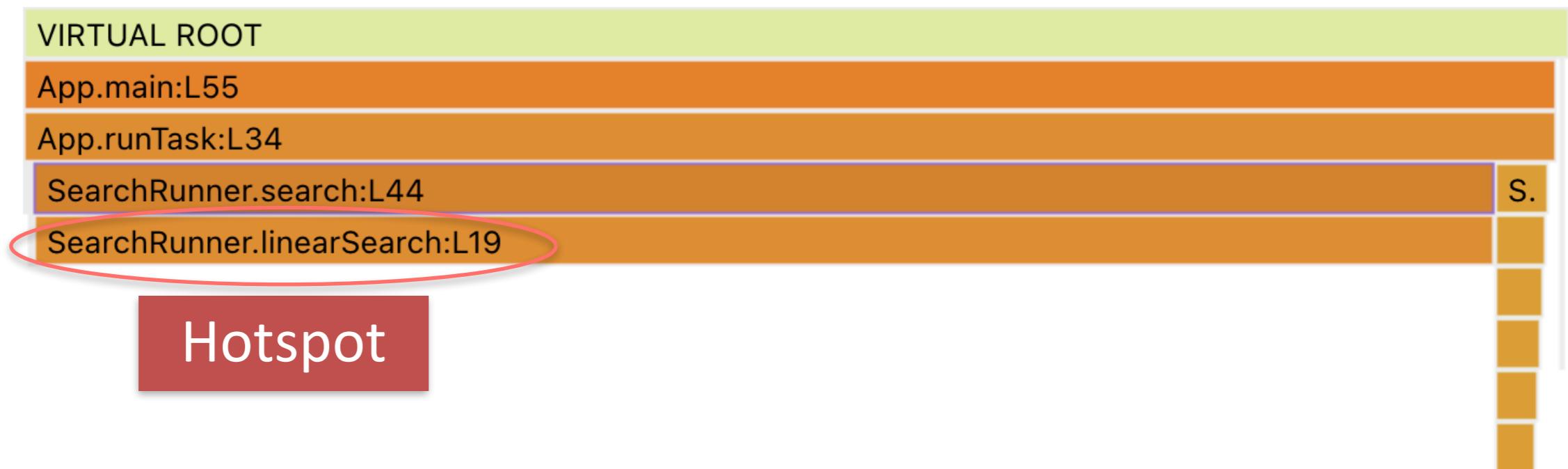
SearchRunner.java

A class provide linearSearch() and binarySearch()

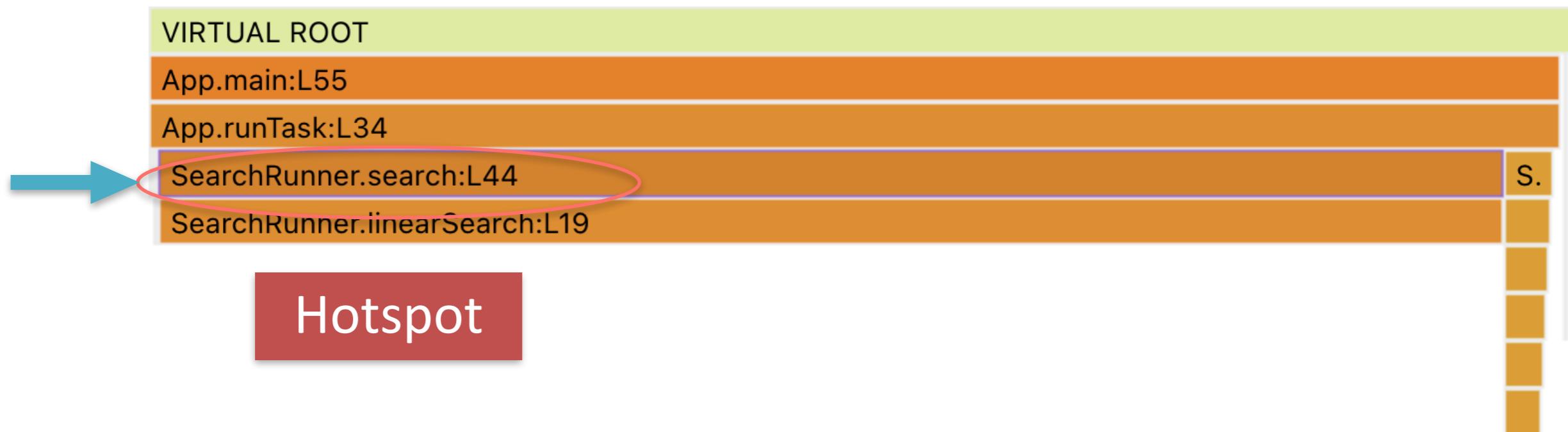
App.java

runttest() first sort an array and then search the array with
linearSearch() function

Case-Directly identify the possible optimization

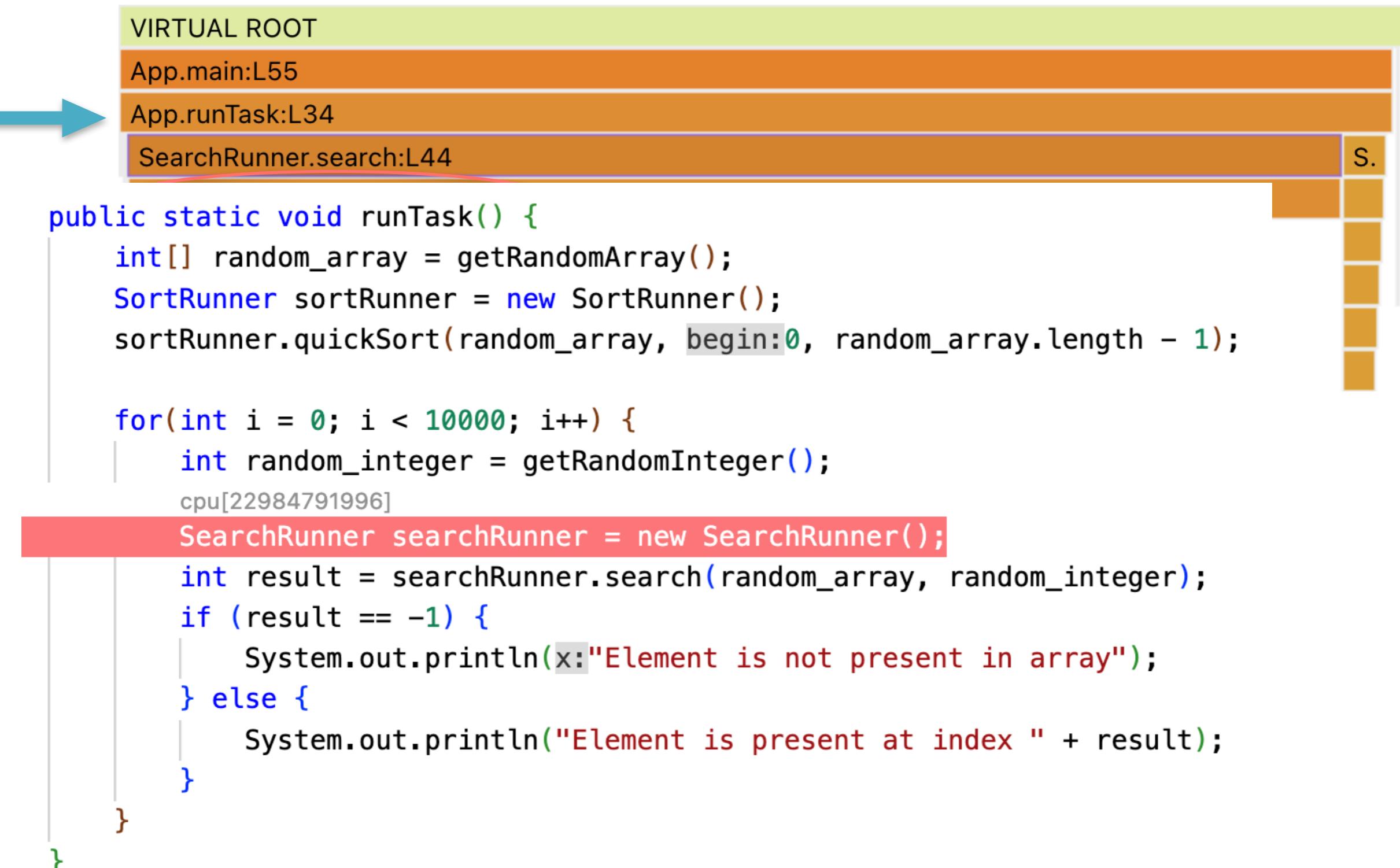


Case-Directly identify the possible optimization

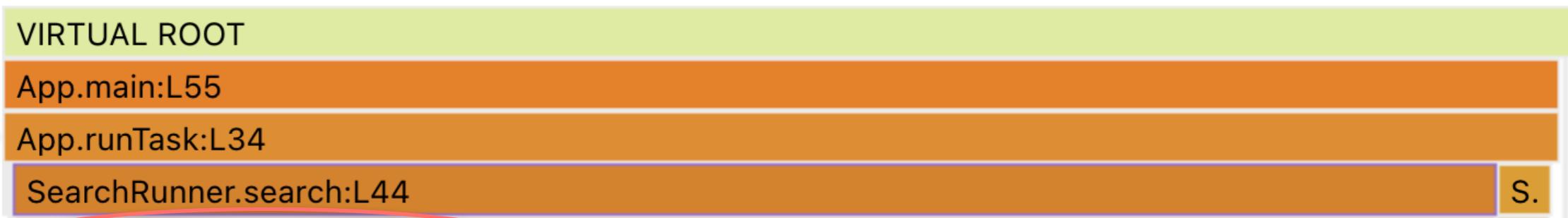


```
public int search(int[] arr, int x) {  
    cpu[21926651379]  
    return linearSearch(arr, x);  
}
```

Case-Directly identify the possible optimization



Case-Directly identify the possible optimization



★

```
public static void runTask() {
    int[] random_array = getRandomArray();
    SortRunner sortRunner = new SortRunner();
    sortRunner.quickSort(random_array, begin:0, random_array.length - 1);

    for(int i = 0; i < 10000; i++) {
        int random_integer = getRandomInteger();
        cpu[22984791996]
        SearchRunner searchRunner = new SearchRunner();
        int result = searchRunner.search(random_array, random_integer);
        if (result == -1) {
            System.out.println("Element is not present in array");
        } else {
            System.out.println("Element is present at index " + result);
        }
    }
}
```

Case-Directly identify the possible optimization

SearchRunner.java — profiles [SSH: (zqd)eb2-3224-lin00.csc.ncsu.edu]

The screenshot shows the Intel VTune Profiler interface. On the left, there's a code editor for `SearchRunner.java` with syntax highlighting for Java. The code implements a binary search algorithm. Below the code editor is a navigation bar with tabs like `profile.ezview` and `micro_bench-cpu`. A toolbar at the bottom includes buttons for `Top Down`, `Bottom Up`, and `Flat`, and a dropdown menu set to `cpu(nanoseconds)`. The main right-hand pane displays a hierarchical performance summary. At the top, it shows a `Code Summary View` with a list of functions and their URLs. Below this is a large tree view where nodes are colored by their execution time: light green for the root node, orange for `App.main`, and yellow for child nodes like `App.runTask` and `SearchRunner.search`. The tree has collapsed some levels, indicated by ellipses.

```
SearchRunner.java 1 ×
javaprofile > java_benchmarks > micro_bench > app > src > main > java > micro_bench > search > SearchRunner.java > {} micro_bench
  31     int pos = lo + (((hi - lo) / (arr[hi] - arr[lo])) * (x - arr[lo]));
  32     if (arr[pos] == x)
  33         return pos;
  34     if (arr[pos] < x)
  35         lo = pos + 1;
  36     else
  37         hi = pos - 1;
  38     }
  39     return -1;
  40 }
  41
  42 public int search(int[] arr, int x) {
  43     cpu[21926651379]
  44     return linearSearch(arr, x);
  45 }
  46
  47

profile.ezview ×
micro_bench-cpu > profile.ezview
...
```

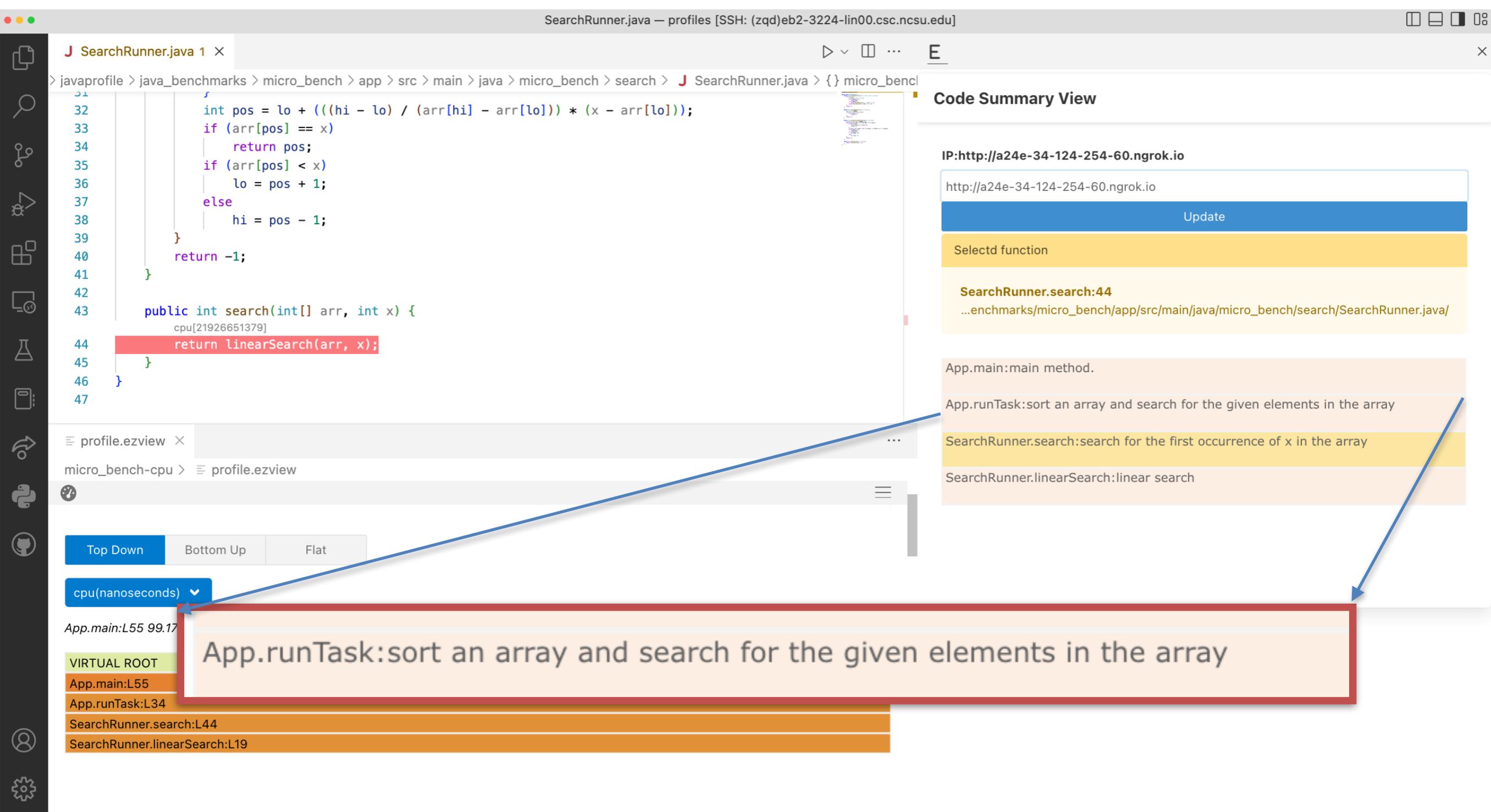
Code Summary View

- IP:<http://a24e-34-124-254-60.ngrok.io>
 - <http://a24e-34-124-254-60.ngrok.io>
 - Update
 - Selectd function
- SearchRunner.search:44**
...enchmarks/micro_bench/app/src/main/java/micro_bench/search/SearchRunner.java/
- App.main:main method.
- App.runTask:sort an array and search for the given elements in the array
- SearchRunner.search:search for the first occurrence of x in the array
- SearchRunner.linearSearch:linear search

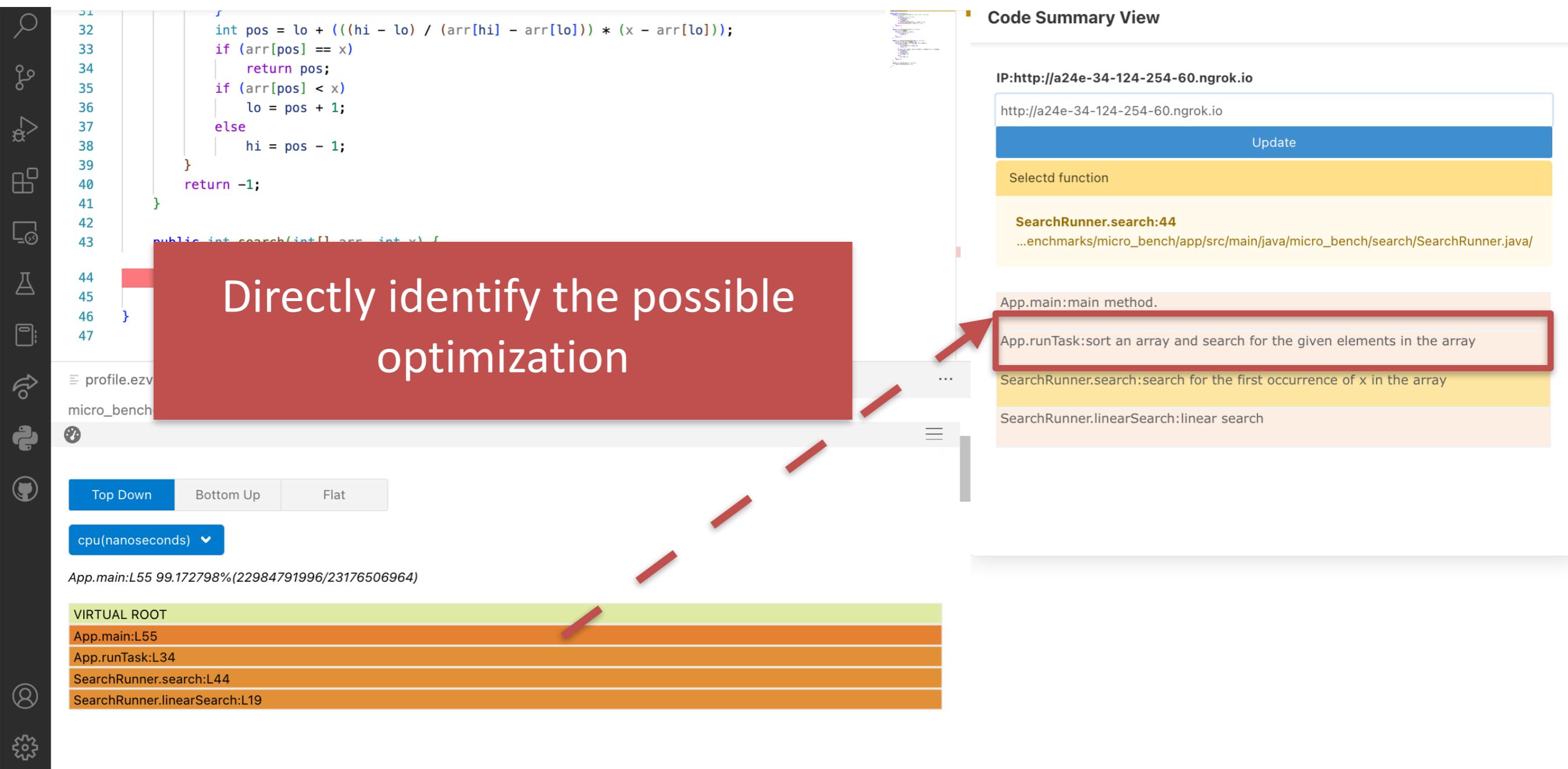
Virtual Root

- VIRTUAL ROOT
- App.main:L55
- App.runTask:L34
- SearchRunner.search:L44
- SearchRunner.linearSearch:L19

Case-Directly identify the possible optimization



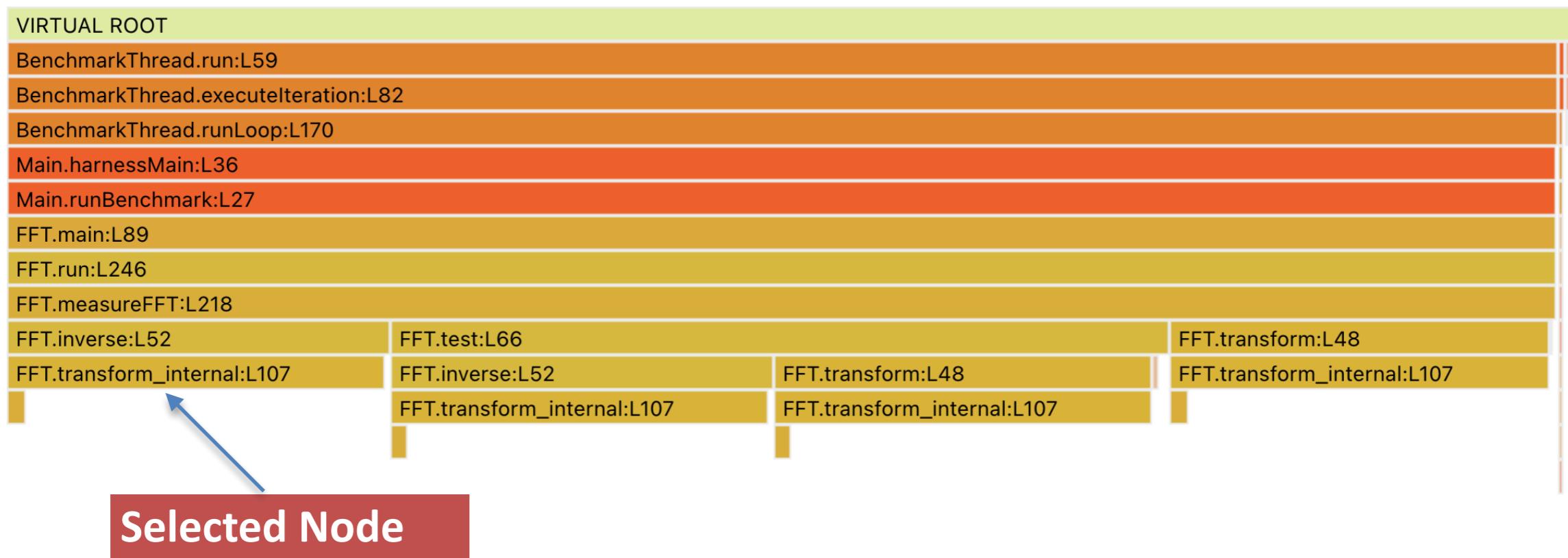
Case-Directly identify the possible optimization



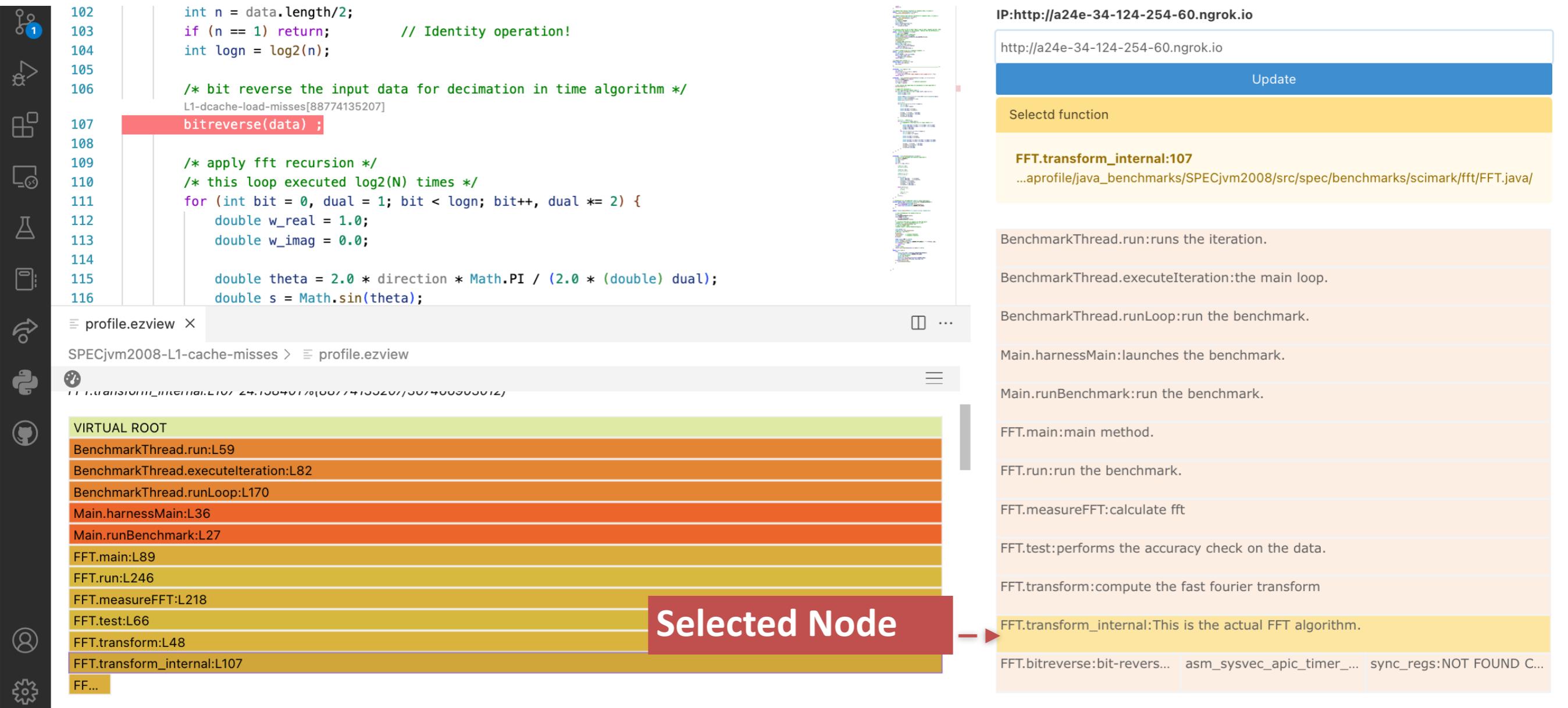
Case-Quickly filter out unsuspicious functions

SPECjvm2008 Scimark.fft

A fast fourier transform benchmark



Case-Quickly filter out unsuspicious functions



Case-Quickly filter out unsuspicious functions

BenchmarkThread.run:runs the iteration.

BenchmarkThread.executeIteration:the main loop.

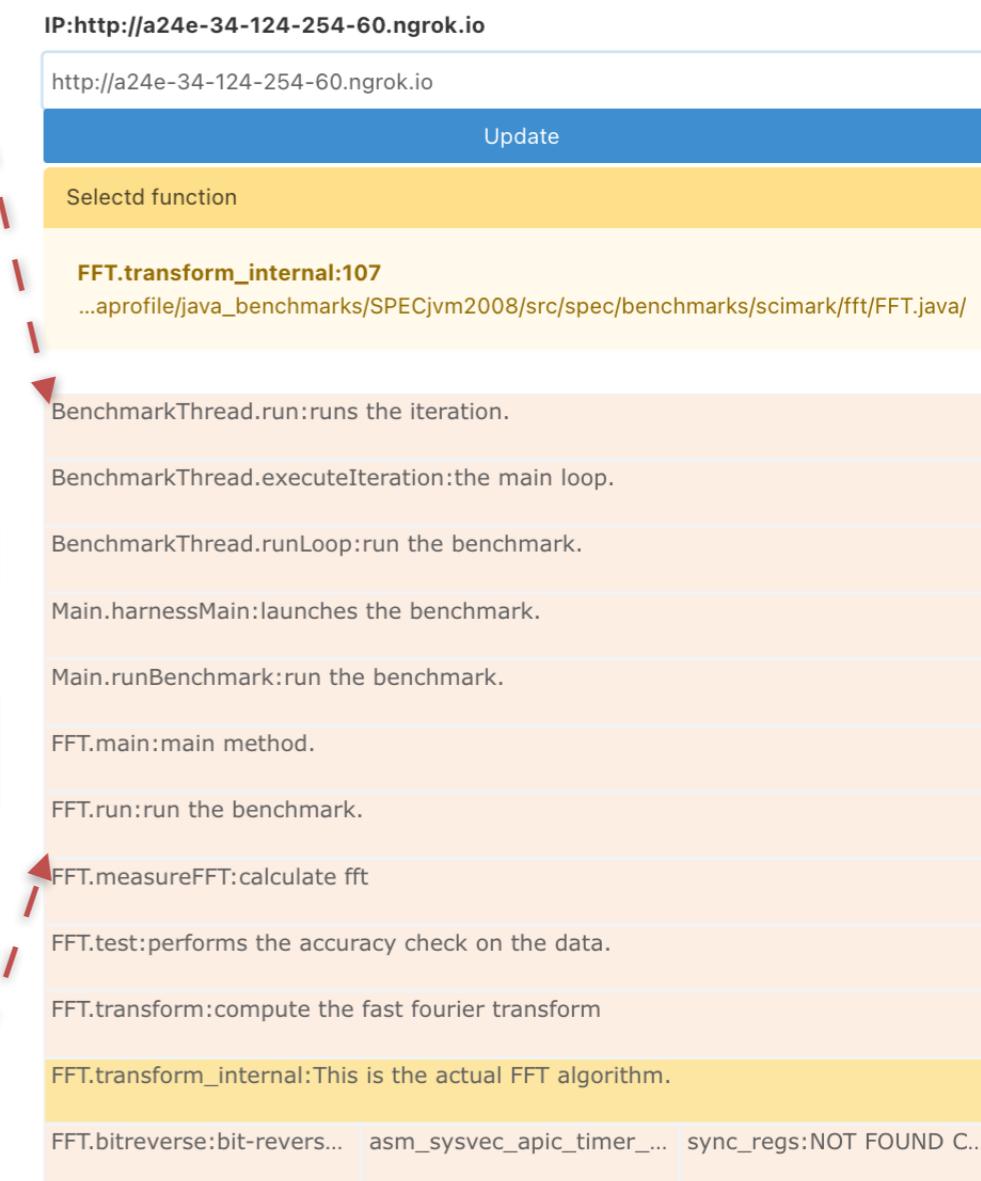
BenchmarkThread.runLoop:run the benchmark.

Main.harnessMain:launches the benchmark.

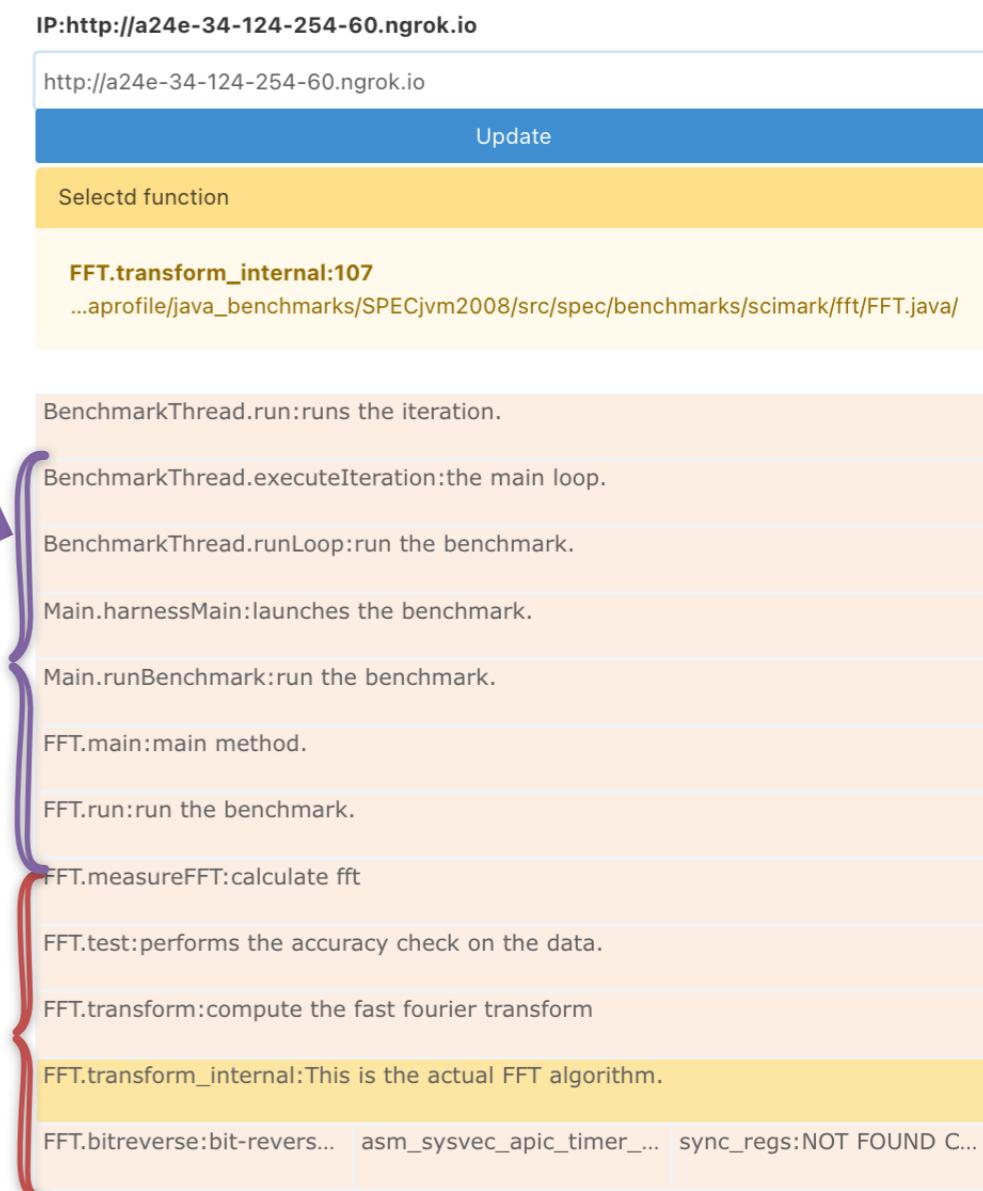
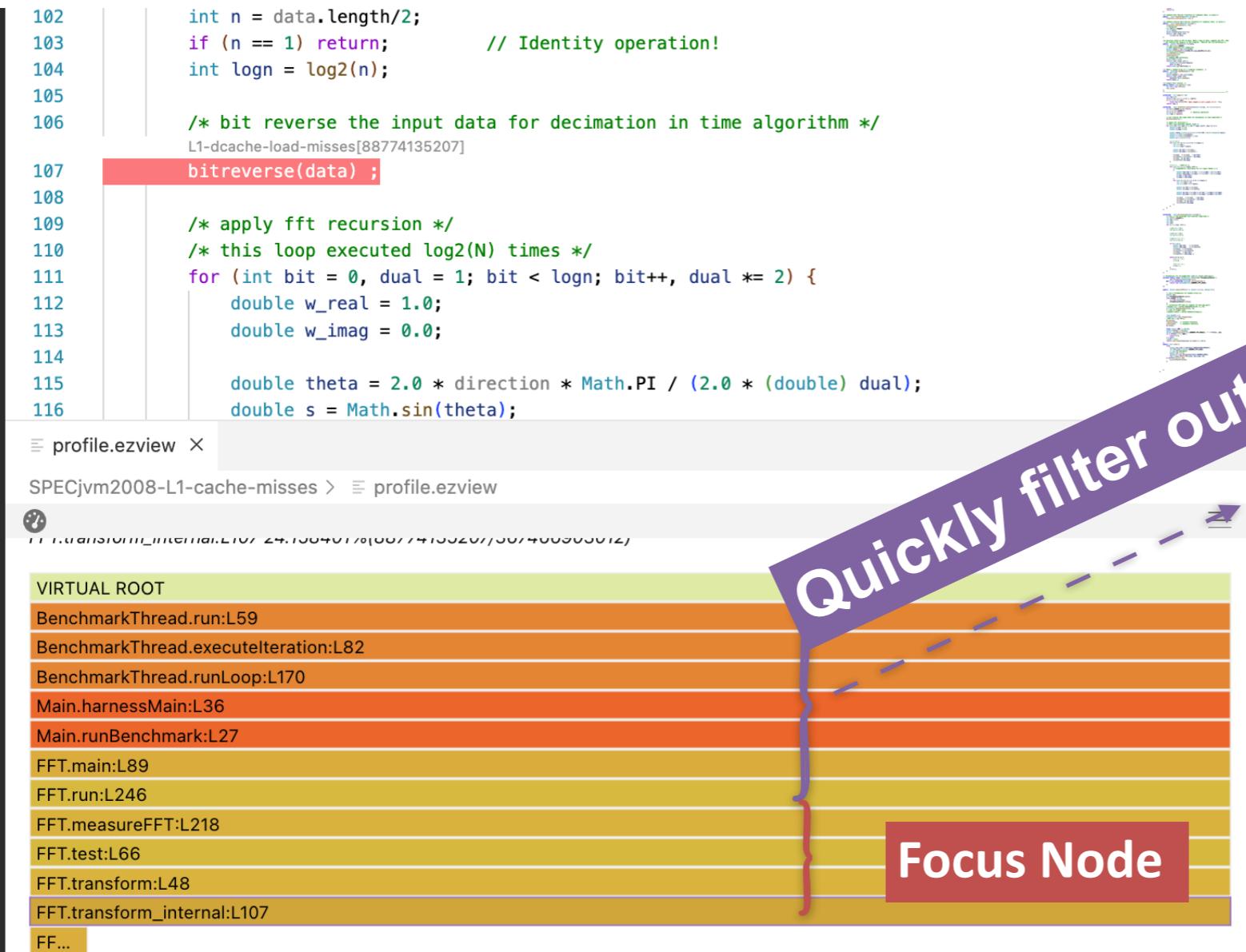
Main.runBenchmark:run the benchmark.

FFT.main:main method.

FFT.run:run the benchmark.



Case-Quickly filter out unsuspicious functions



Conclusion

DeepProf (An extension for EasyView)

- Supports Async-Profiler and other profilers capable of generating output in pprof format.
- A new code summary model (opt for call path summary)
- A new code summary view

Show cases that utilize the code summary view to speedup finding performance issues.

Future Work

Support the analysis of code summary

Support light-wight model

Try to use LLM to give auto fix suggestions

Thank you!