

NC STATE

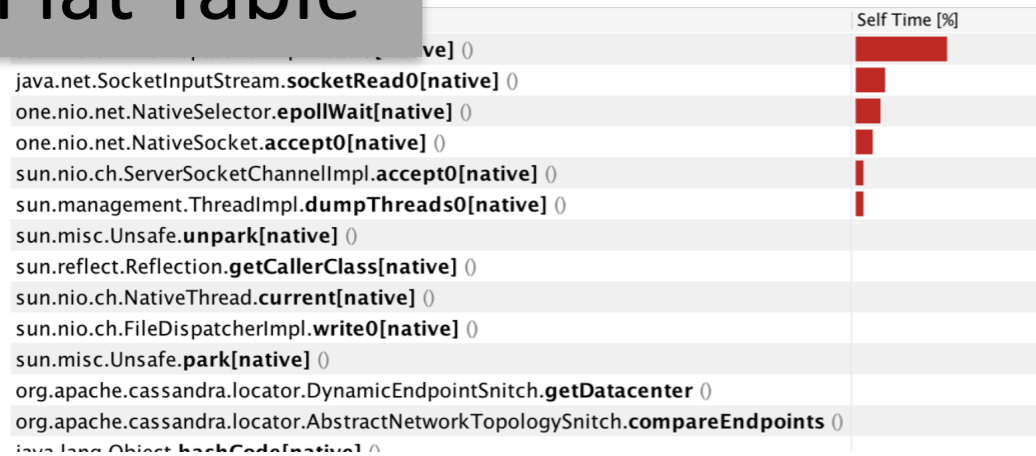
DeepProf: Interpreting Performance Profiles with Deep Learning

Qidong Zhao (Advisor: Dr. Xu Liu)
qzhao24@ncsu.edu
North Carolina State University

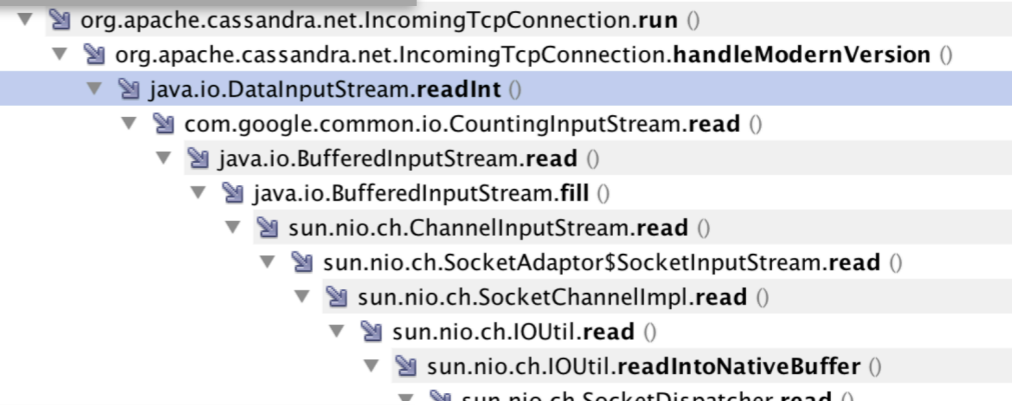
Scalable Tools Workshop
Granlibakken, Lake Tahoe, California
June 20 2023

Exploring the Visualizations of Profiles

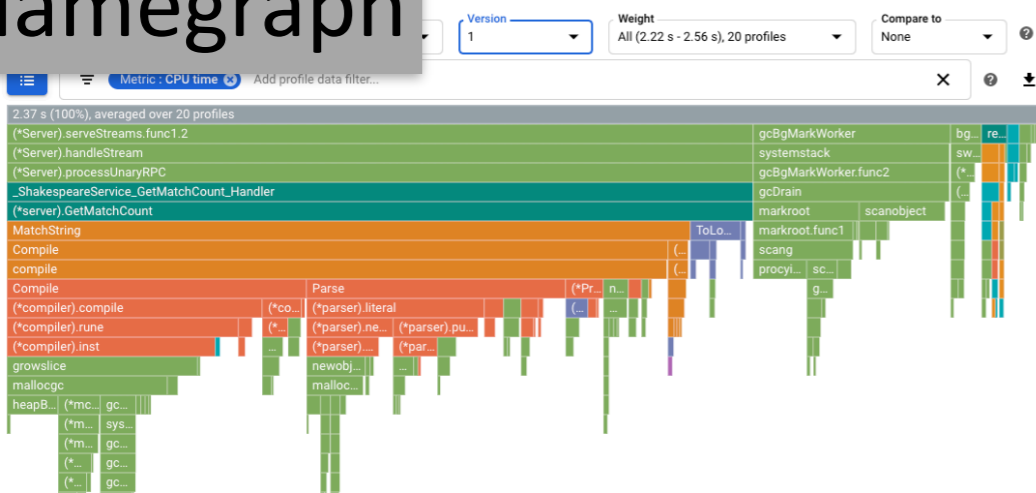
Flat Table



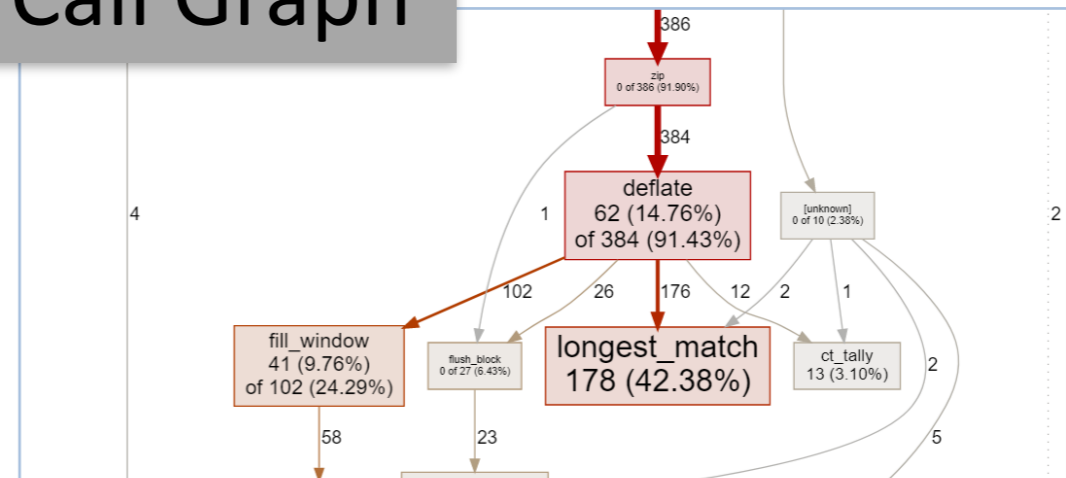
Tree Table



Flamegraph

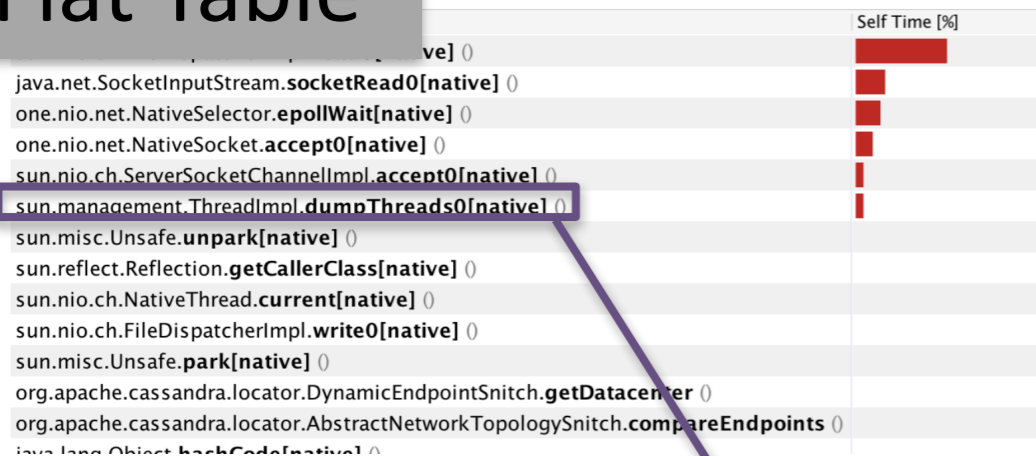


Call Graph

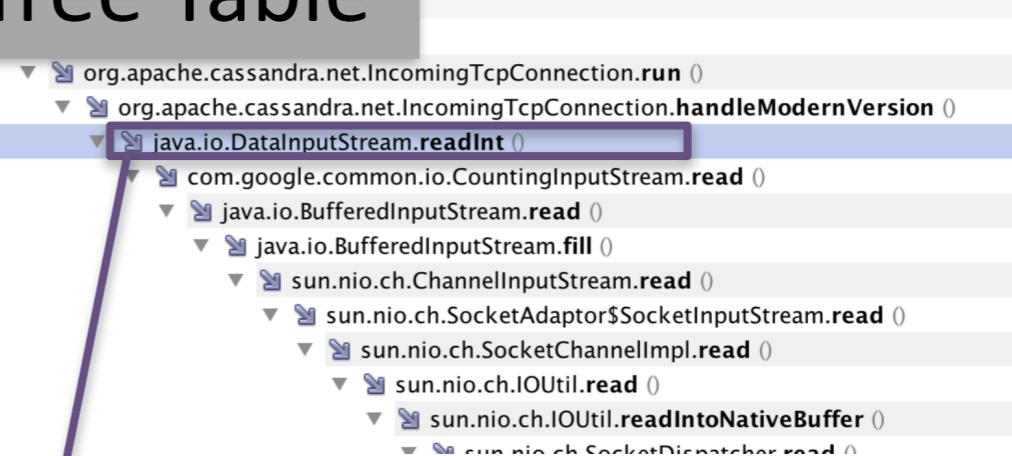


Exploring the Visualizations of Profiles

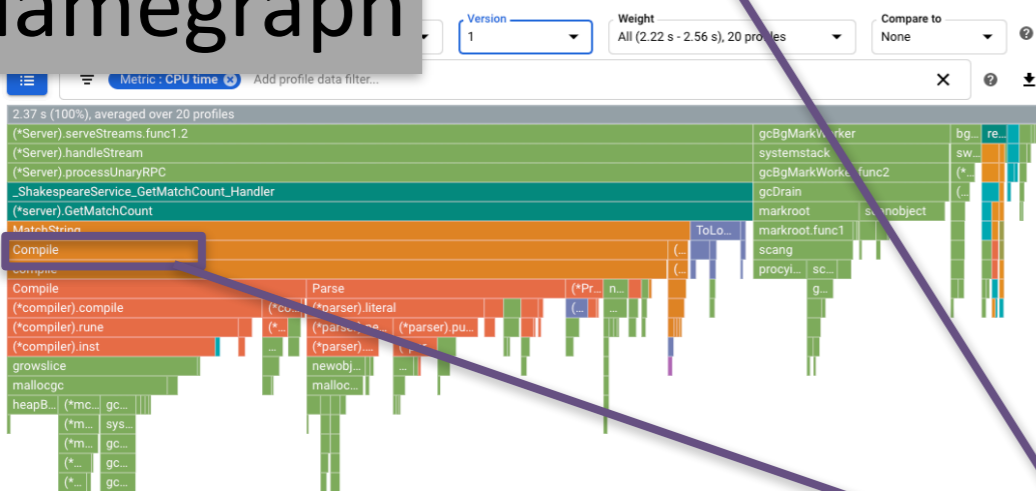
Flat Table



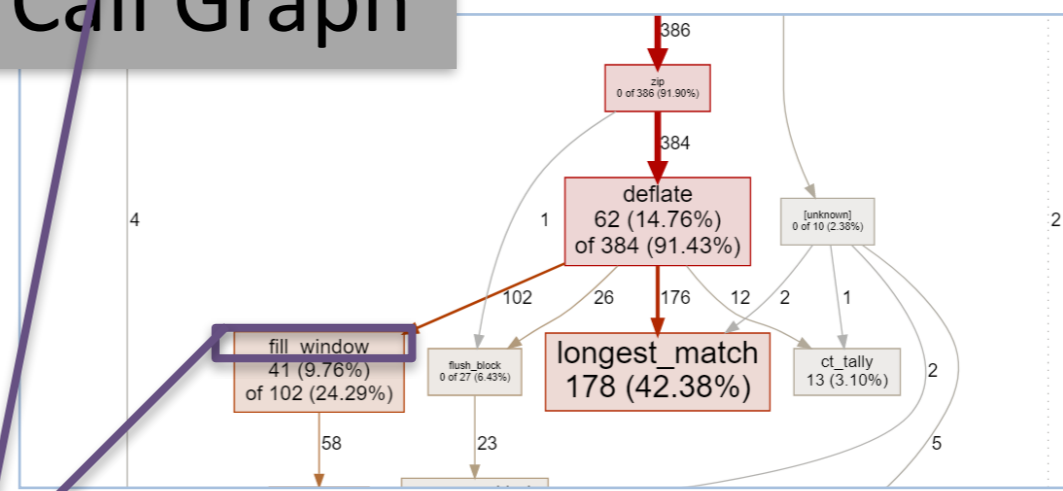
Tree Table



Flamegraph

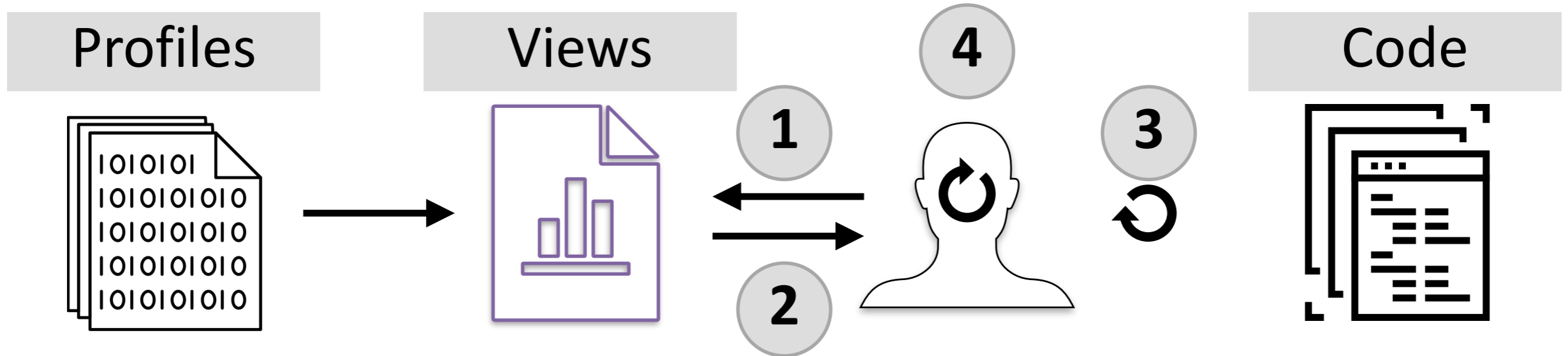


Call Graph



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



4 Determine whether func can be optimized

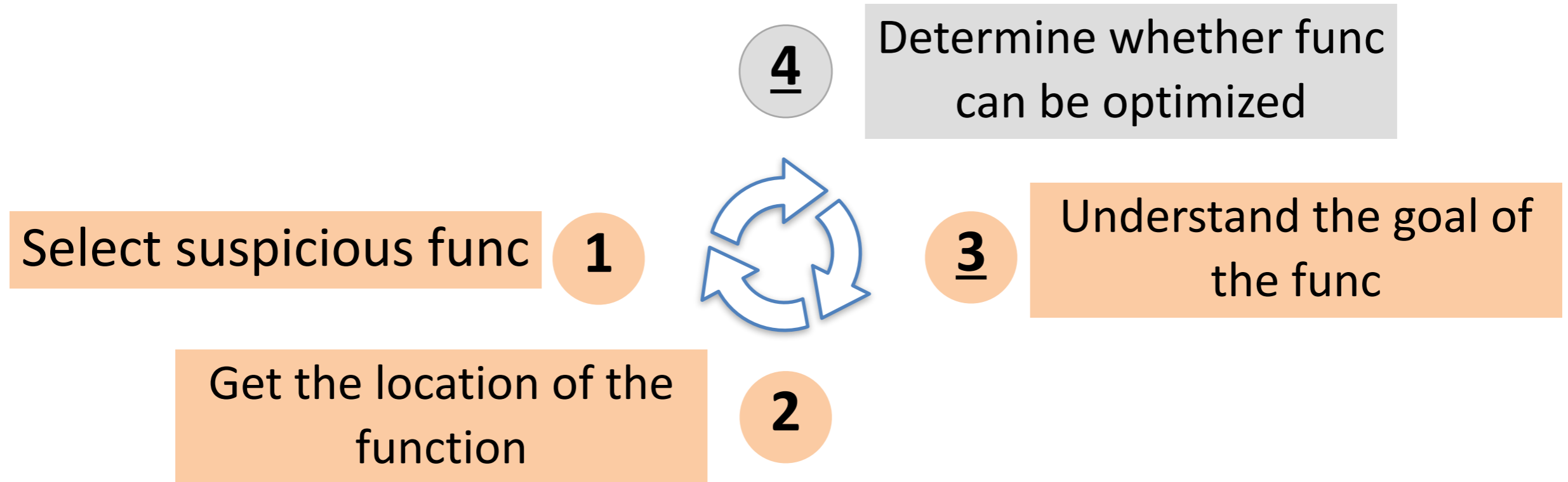
Select suspicious func 1

3 Understand the goal of the func

Get the location of the function 2

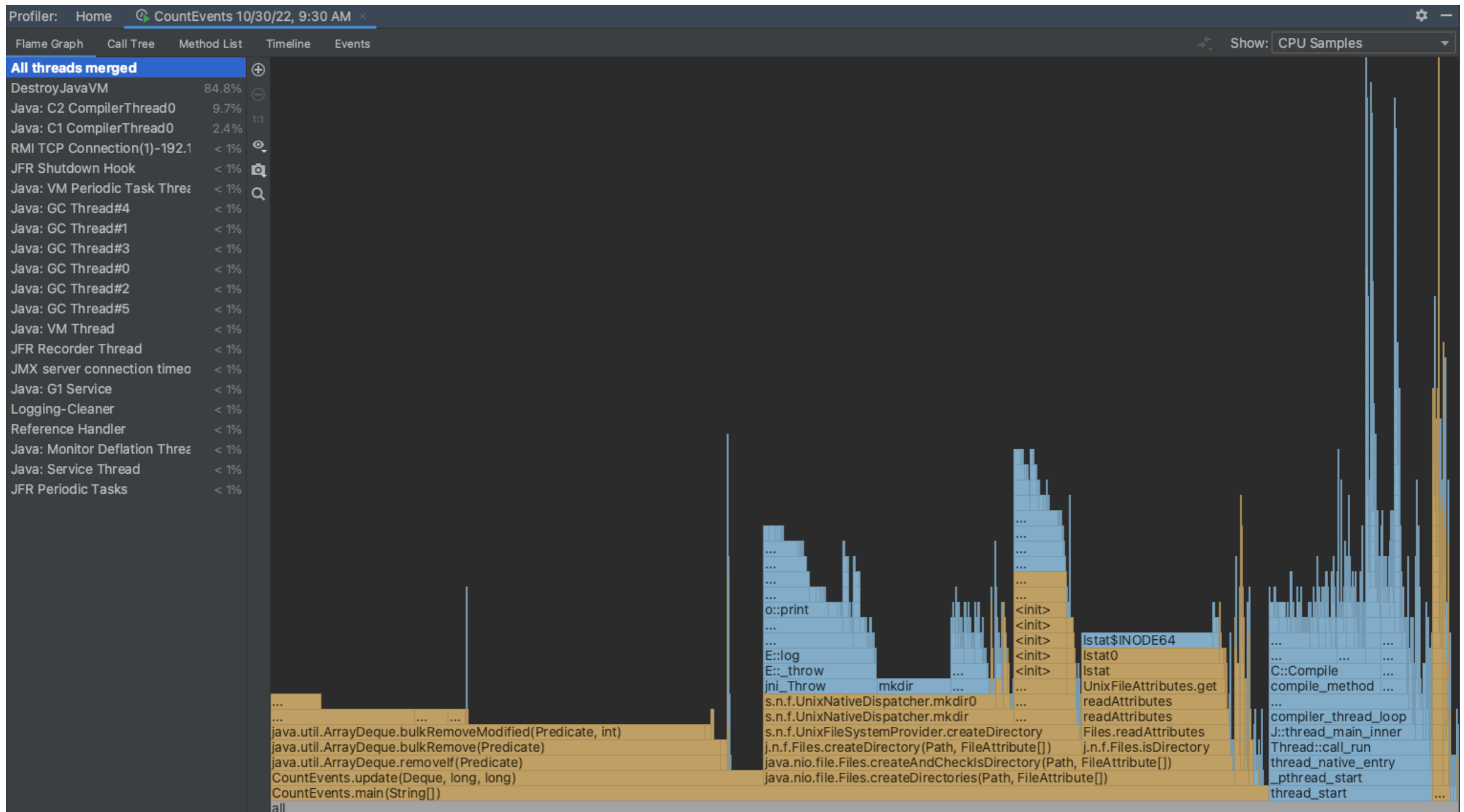


Challenges in Interpreting Profiles



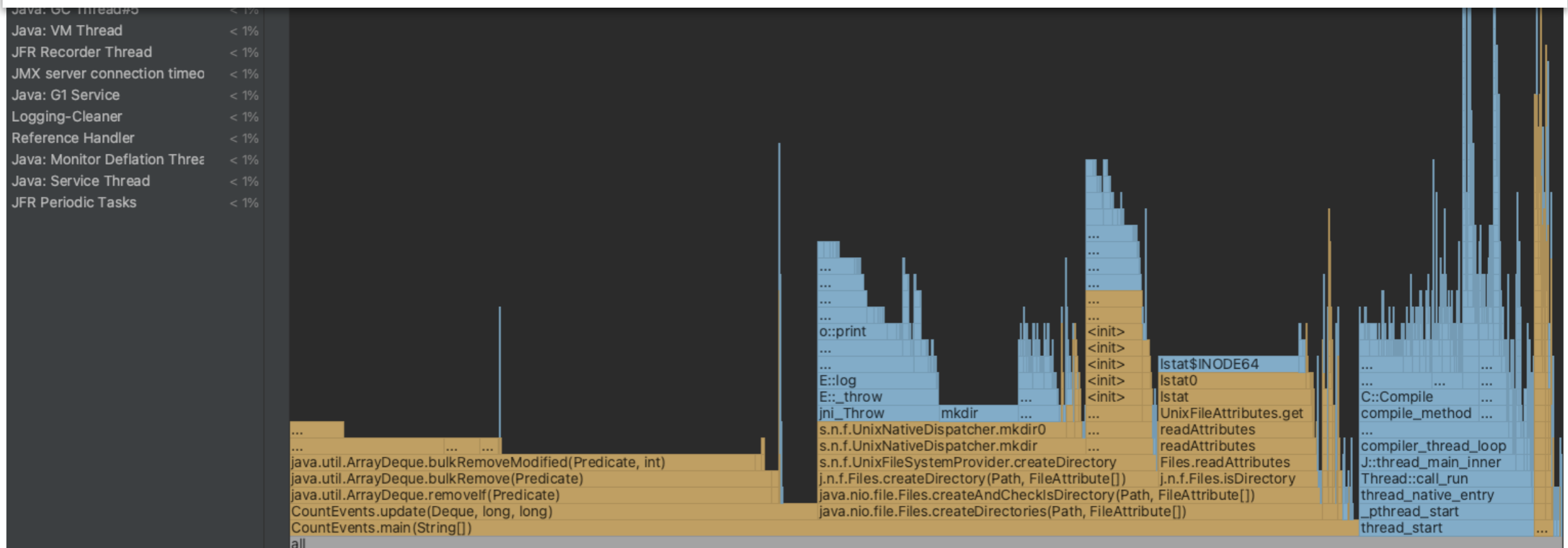
- 1 ✘ Not able to understand a function's goal from its name
- 2 ✘ Missing information for mapping to source code
- 3 ✘ Not easy to understand the code of complex applications

Profile generated by Async-Profiler

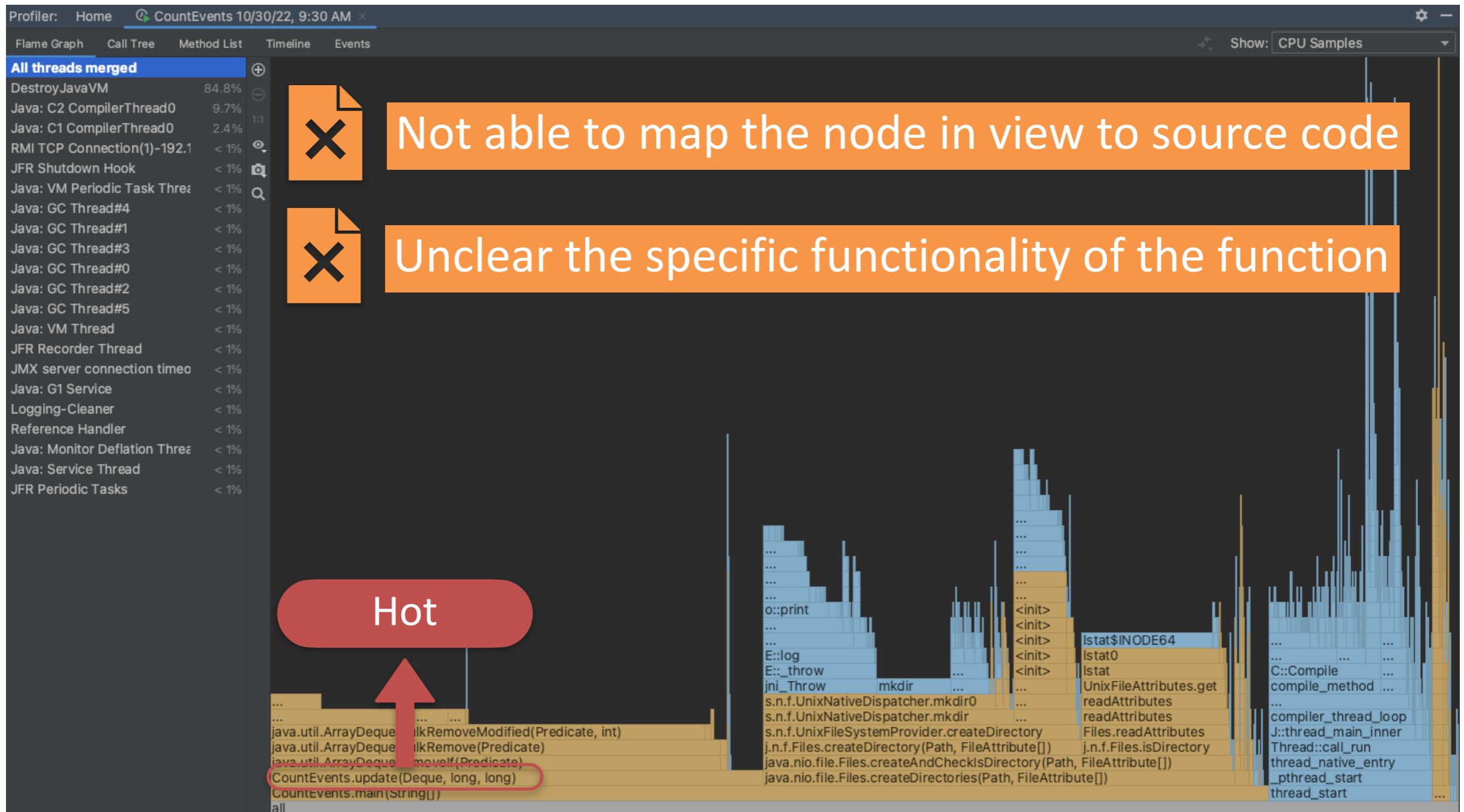


Profile generated by Async-Profiler

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



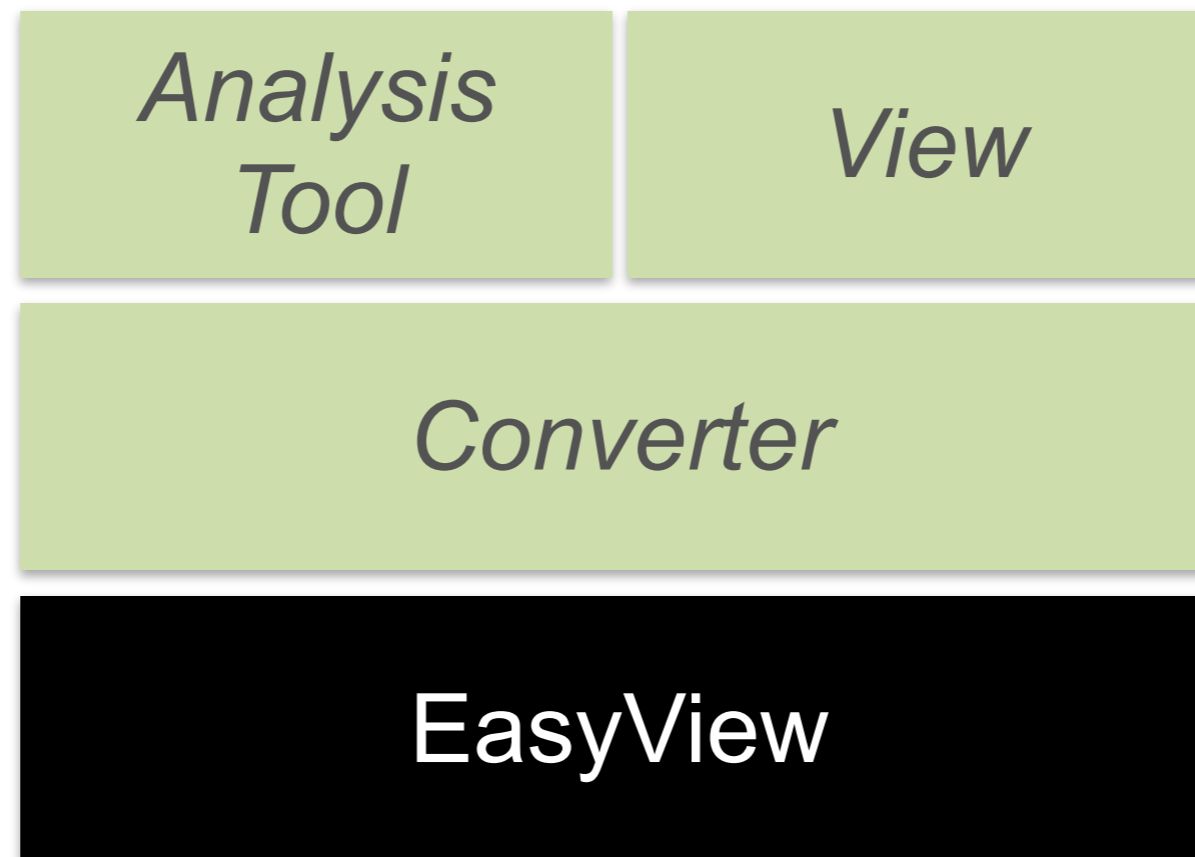
Profile generated by Async-Profiler



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**

The screenshot displays the EasyView interface integrated with a code editor. The top part shows two code files: `http_util.go` and `passthrough.go`. The `passthrough.go` file has a function `start` at line 48, which is highlighted with a green circle and a callout box containing an "EasyView Tip" and a list of memory addresses. Below the code editor is the EasyView visualization window, which shows a hierarchical tree of nodes representing the execution flow. The nodes are color-coded and include details such as function names and line numbers. A callout box with a green checkmark and a bar chart highlights a specific node: `passthrough.(*passthroughResolver).start:L48 ...`. The visualization also includes a "Show Metric" dropdown set to `alloc_space(bytes)` and a "VIRTUAL ROOT" section at the top.

```
610 func newBufWriter(conn net.Conn, batchSize int) *bufWriter
611     return &bufWriter{
612         alloc_space[38409218780]
613         buf:      make([]byte, batchSize*2),
614         batchSize: batchSize,
615         conn:     conn,
616     }
617 }
```

```
46
47 func (r *passthroughResolver) start(
48     r.cc.UpdateState(resolver.State{Addresses: []resolver.Address
49 }
50
51 func (*passthroughResolver) ResolveNow(o resolver.ResolveNowO
```

memory.ezview U X

Top Down Bottom Up Flat Show Metric: alloc_space(bytes)

VIRTUAL ROOT

- grpc.(*addrConn).resetTransport:L1118
- grpc.(*addrConn).tryAllAddrs:L1208
- transport.NewClientTransport:L581
- transport.newHTTP2Client:L249
- bufio.NewReaderSize:L57
- transport.newBufWriter:L612

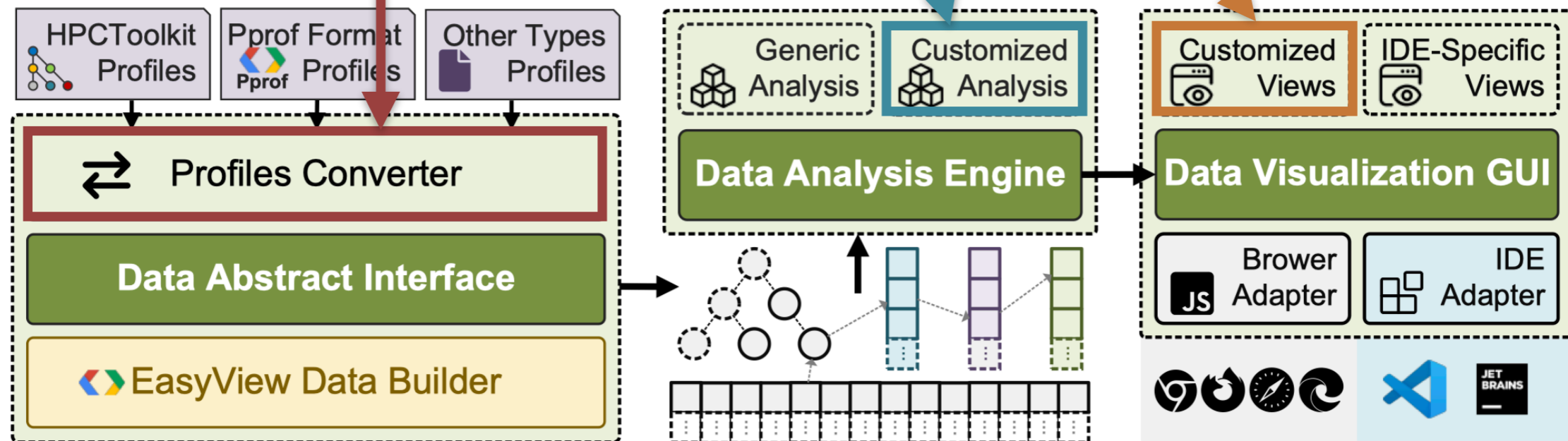
transport.newBufWriter:L612 36.27%(384092...
Tip: right click node to open source file

passthrough.(*passthroughResolver).start:L48 ...
Tip: right click node to open source file

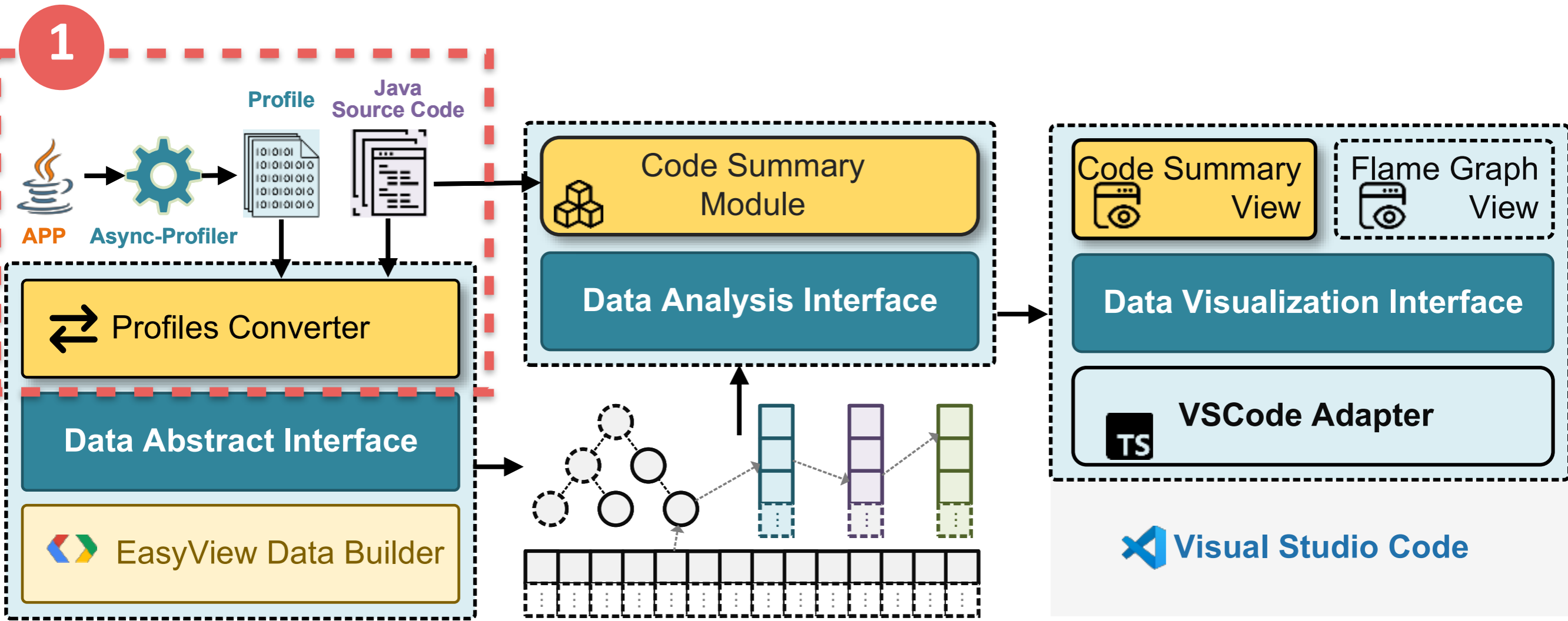
Background-EasyView

Providing interfaces that allow creating

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



DeepProf

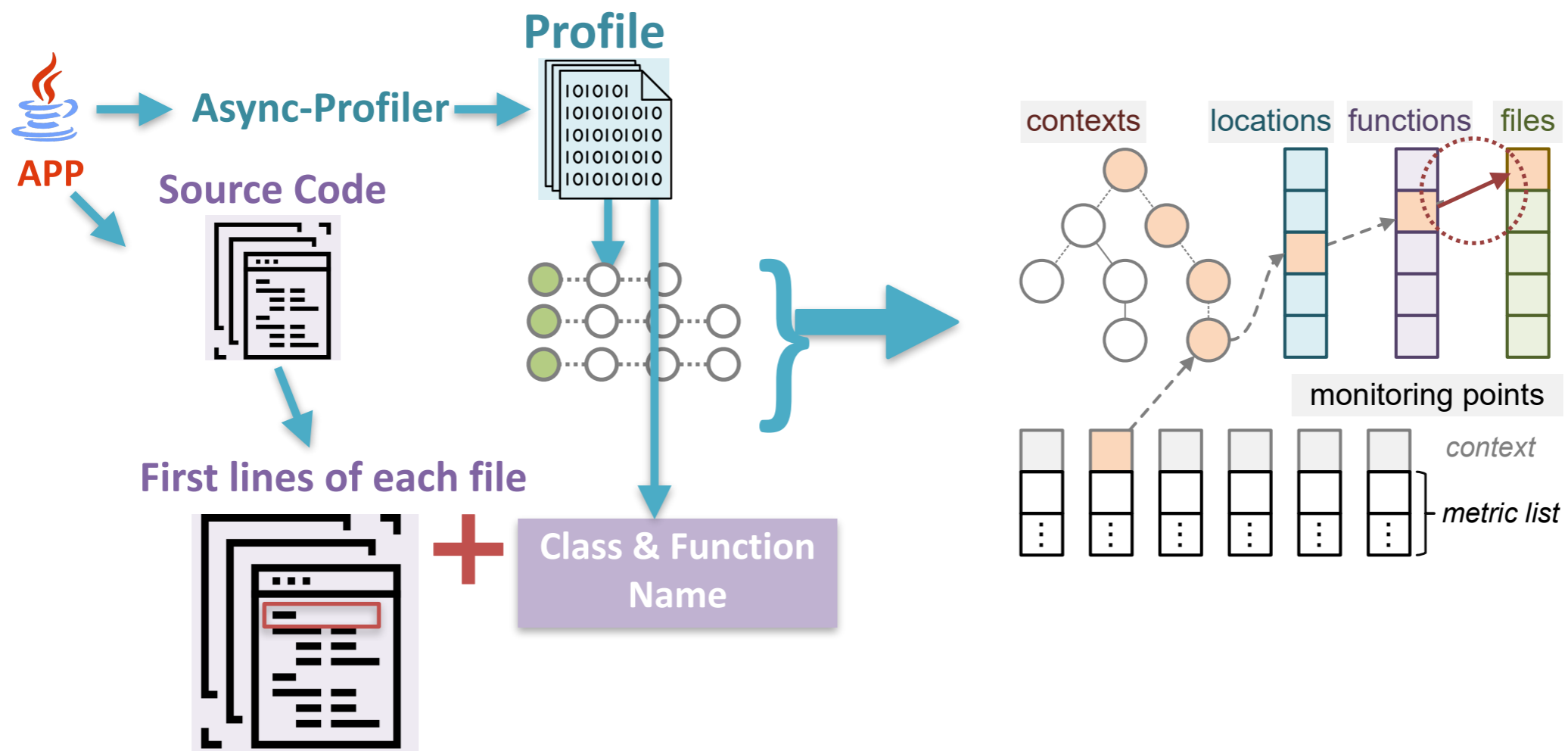


Missing information for mapping to source file

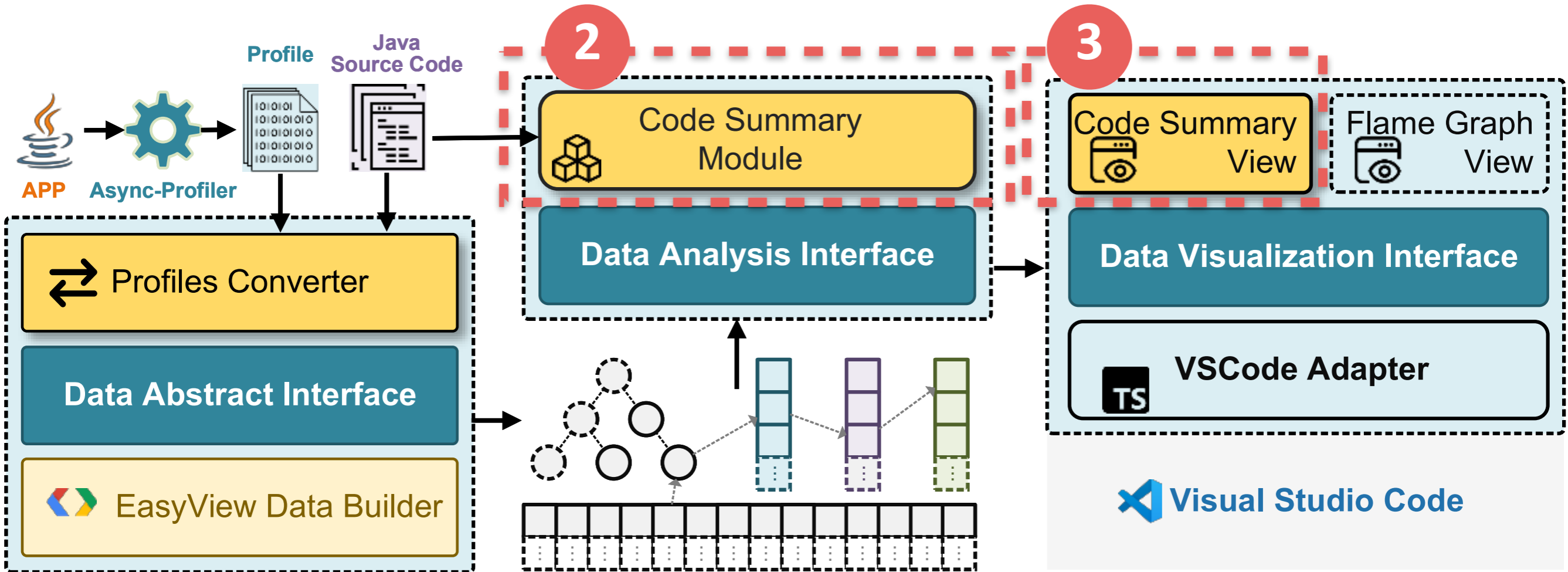
1. Automatically Enhance the Profiles

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

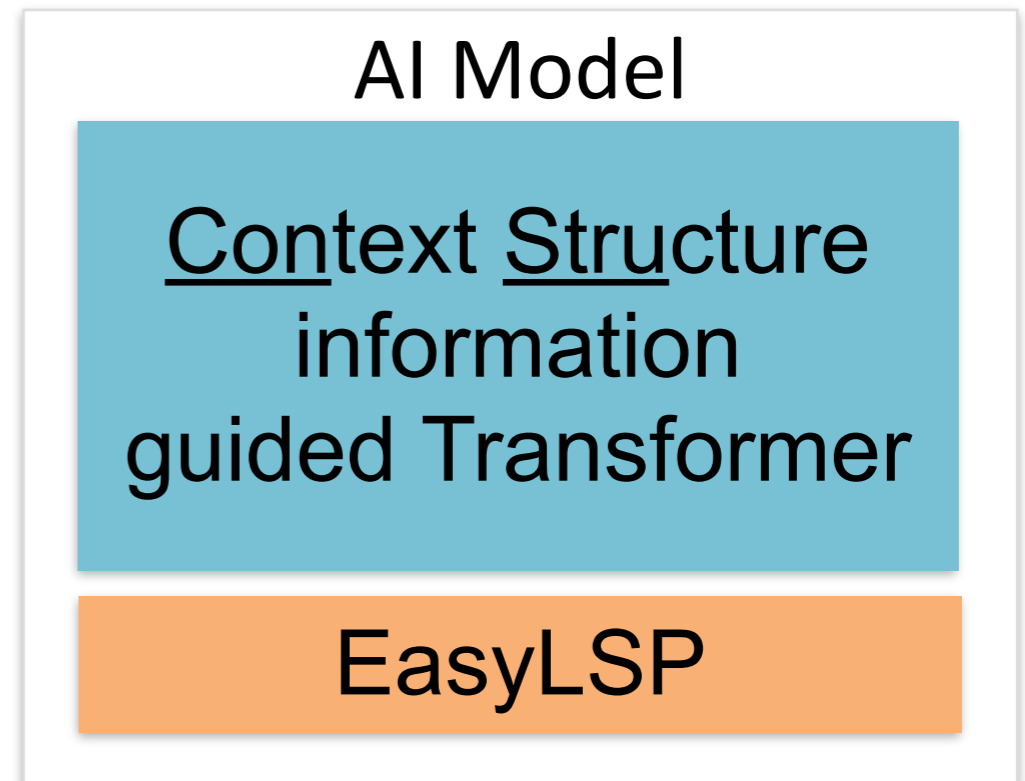
Model Architecture

The state of the art model

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

Our code summary model

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



Code Summary Model

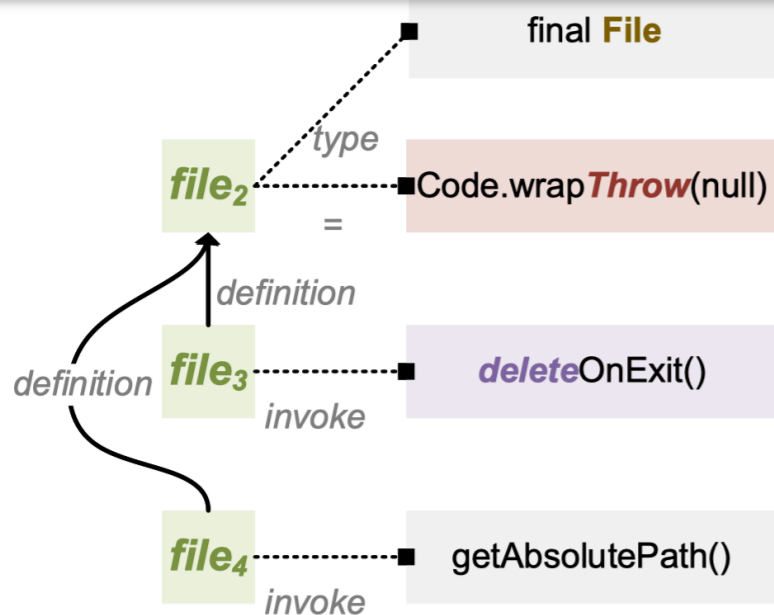
EasyLSP

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

```

1: public static String tmpJournal() {
2:   final File file = Code.wrapThrow(null);
3:   file.deleteOnExit();
4:   return file.getAbsolutePath();}
    
```

Source Code



Language Server Protocol

EasyLSP

	<i>file₂</i>	<i>throw₂</i>	<i>file₃</i>	<i>delete₃</i>	<i>file₄</i>
<i>file₂</i>	1	1	1	1	1
<i>throw₂</i>	1	1	0.33	0	0.33
<i>file₃</i>	1	0.33	1	1	1
<i>delete₃</i>	1	0	1	1	0.33
<i>file₄</i>	1	0.33	1	0.33	1

Position Matrix

	...	<i>file₂</i>	...	<i>throw₂</i>	...	<i>file₃</i>	<i>delete₃</i>	...	<i>file₄</i>	...
<i>file₂</i>	...	1	...	1	...	1	1	...	1	...
<i>throw₂</i>	...	1	...	1	...	1	0	...	1	...
<i>file₃</i>	...	1	...	1	...	1	1	...	1	...
<i>delete₃</i>	...	1	...	0	...	1	1	...	1	...
<i>file₄</i>	...	1	...	1	...	1	1	...	1	...

LSP Matrix

Context Structure information guided Transformer

ConStruct

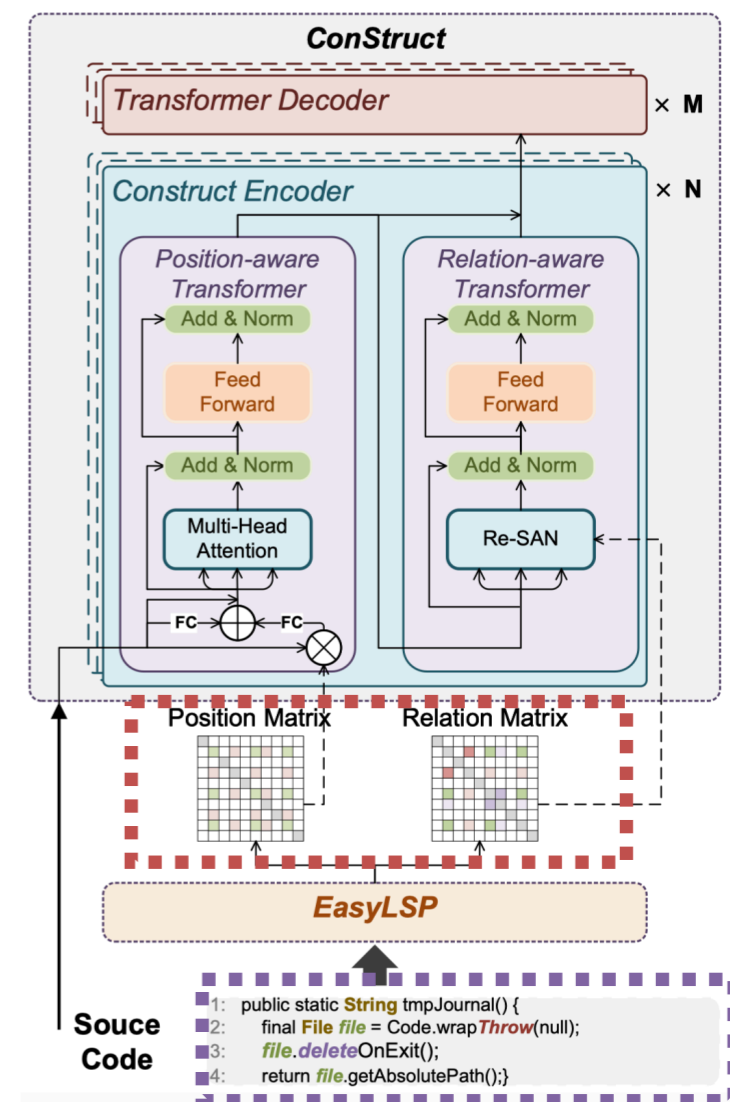
a context structure information guided transformer

(1) A novel attention mechanism

(2) A new encode

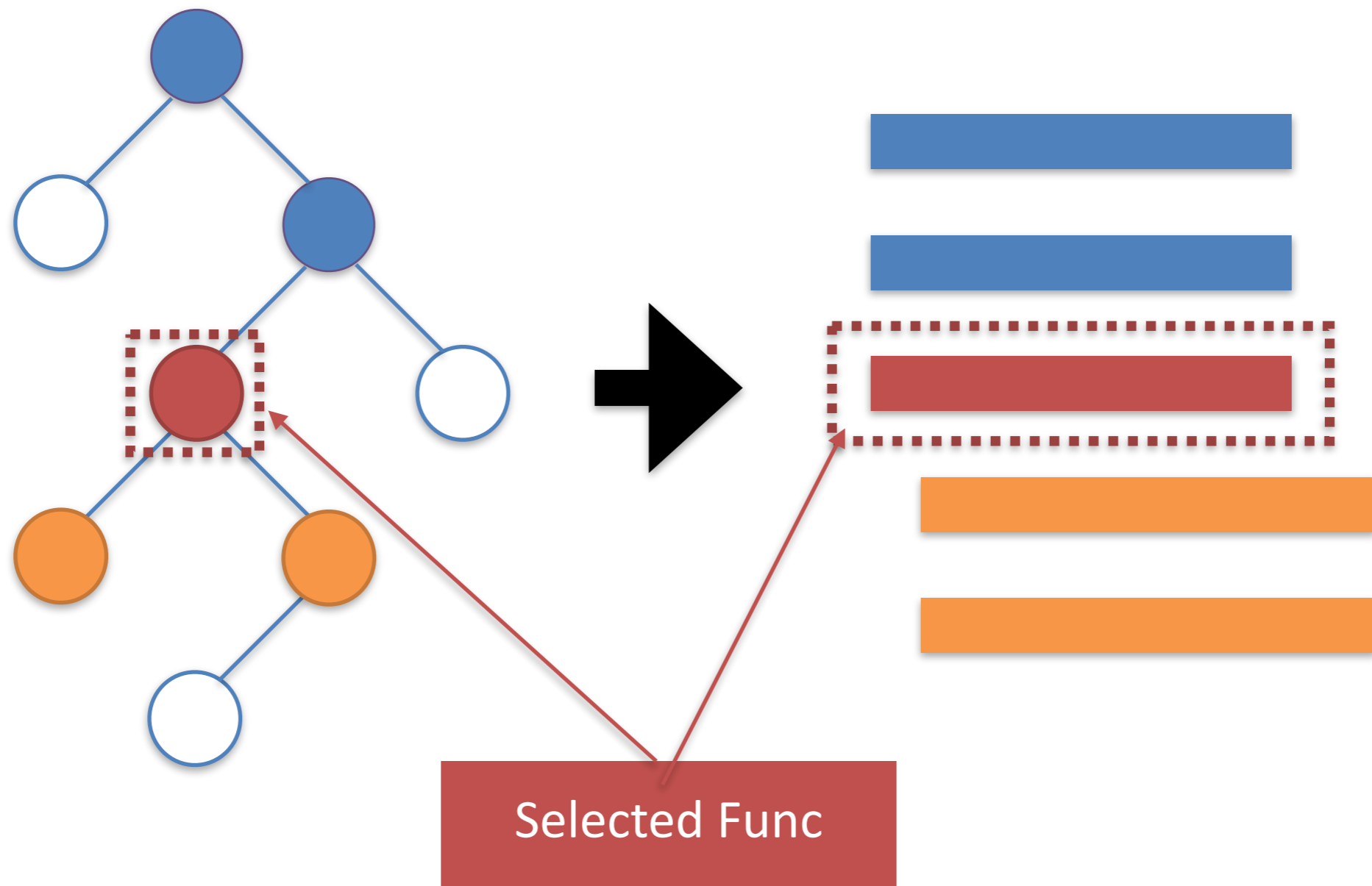
- sequential code inputs
- relation matrix
- position matrix

(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()

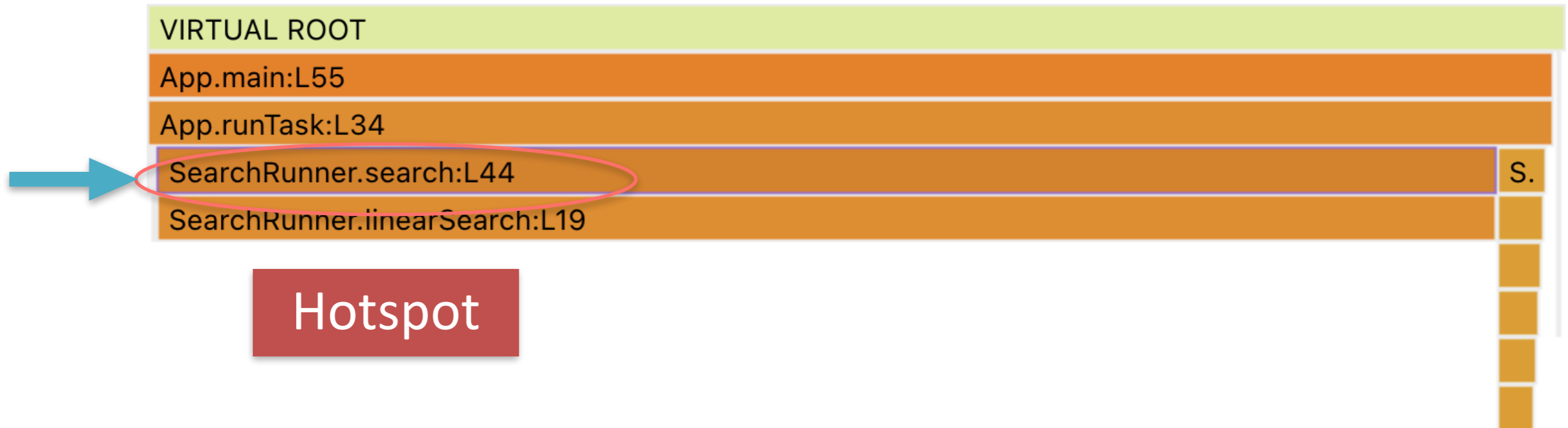
SearchRunner.java

A class provide linearSearch() and binarySearch()

App.java

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

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

VIRTUAL ROOT

App.main:L55

App.runTask:L34

SearchRunner.search:L44

S.

```
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(x:"Element is not present in array");
        } else {
            System.out.println("Element is present at index " + result);
        }
    }
}
```


Case-Directly identify the possible optimization



VIRTUAL ROOT

App.main:L55

App.runTask:L34

SearchRunner.search:L44

S.



```
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();  
        SearchRunner searchRunner = new SearchRunner();  
        int result = searchRunner.search(random_array, random_integer);  
        if (result == -1) {  
            System.out.println(x:"Element is not present in array");  
        } else {  
            System.out.println("Element is present at index " + result);  
        }  
    }  
}
```

Case-Directly identify the possible optimization

The screenshot displays an IDE window for `SearchRunner.java` with the following code:

```
31  
32     int pos = lo + (((hi - lo) / (arr[hi] - arr[lo])) * (x - arr[lo]));  
33     if (arr[pos] == x)  
34         return pos;  
35     if (arr[pos] < x)  
36         lo = pos + 1;  
37     else  
38         hi = pos - 1;  
39 }  
40 return -1;  
41 }  
42  
43 public int search(int[] arr, int x) {  
44     return linearSearch(arr, x);  
45 }  
46 }  
47
```

The IDE also shows a **Code Summary View** on the right side, which includes:

- IP: `http://a24e-34-124-254-60.ngrok.io`
- Search input: `http://a24e-34-124-254-60.ngrok.io`
- Update button
- 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

At the bottom, the IDE shows a performance profile for `cpu(nanoseconds)` with the following data:

`App.main:L55 99.172798%(22984791996/23176506964)`

Method	Line
VIRTUAL ROOT	
App.main	L55
App.runTask	L34
SearchRunner.search	L44
SearchRunner.linearSearch	L19

Case-Directly identify the possible optimization

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

J SearchRunner.java 1 x

```

32     int pos = lo + (((hi - lo) / (arr[hi] - arr[lo])) * (x - arr[lo]));
33     if (arr[pos] == x)
34         return pos;
35     if (arr[pos] < x)
36         lo = pos + 1;
37     else
38         hi = pos - 1;
39 }
40 return -1;
41 }
42
43 public int search(int[] arr, int x) {
44     return linearSearch(arr, x);
45 }
46 }
47

```

profile.ezview x

micro_bench-cpu > profile.ezview

Top Down Bottom Up Flat

cpu(nanoseconds) v

App.main:L55 99.17

VIRTUAL ROOT

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

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

App.runTask:sort an array and search for the given elements in the array

Case-Directly identify the possible optimization

Directly identify the possible optimization

```
31
32     int pos = lo + (((hi - lo) / (arr[hi] - arr[lo])) * (x - arr[lo]));
33     if (arr[pos] == x)
34         return pos;
35     if (arr[pos] < x)
36         lo = pos + 1;
37     else
38         hi = pos - 1;
39 }
40 return -1;
41 }
42
43 public int search(int[] arr, int x) {
44
45 }
46 }
47 }
```

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

profile.ezv
micro_bench

Top Down Bottom Up Flat

cpu(nanoseconds)

App.main:L55 99.172798%(22984791996/23176506964)

VIRTUAL ROOT

App.main:L55

App.runTask:L34

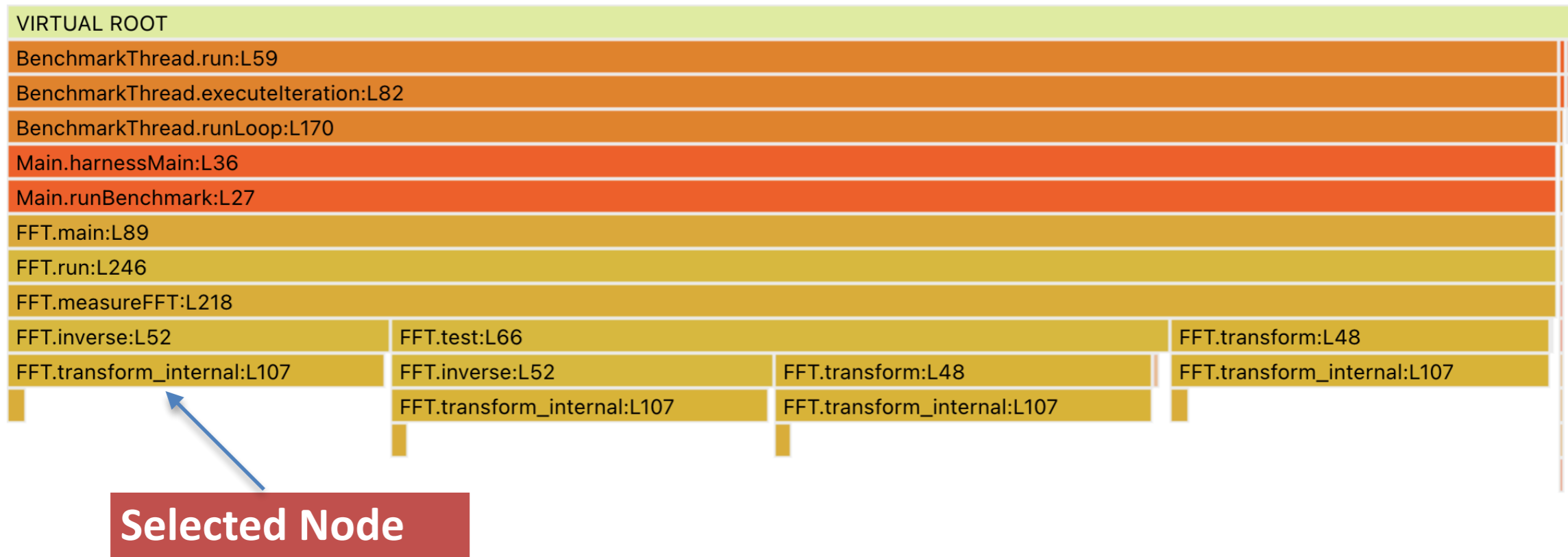
SearchRunner.search:L44

SearchRunner.linearSearch:L19

Case-Quickly filter out unsuspecting functions

SPECjvm2008 Scimark.fft

A fast fourier transform benchmark



Case-Quickly filter out unsuspecting functions

```

102     int n = data.length/2;
103     if (n == 1) return;           // Identity operation!
104     int logn = log2(n);
105
106     /* bit reverse the input data for decimation in time algorithm */
107     bitreverse(data);
108
109     /* apply fft recursion */
110     /* this loop executed log2(N) times */
111     for (int bit = 0, dual = 1; bit < logn; bit++, dual *= 2) {
112         double w_real = 1.0;
113         double w_imag = 0.0;
114
115         double theta = 2.0 * direction * Math.PI / (2.0 * (double) dual);
116         double s = Math.sin(theta);
    
```

profile.ezview ×

SPECjvm2008-L1-cache-misses > profile.ezview

FF...transform_internal:L107 24.100401% (88774135207/36740090012)

- VIRTUAL ROOT
- BenchmarkThread.run:L59
- BenchmarkThread.executeIteration:L82
- BenchmarkThread.runLoop:L170
- Main.harnessMain:L36
- Main.runBenchmark:L27
- FFT.main:L89
- FFT.run:L246
- FFT.measureFFT:L218
- FFT.test:L66
- FFT.transform:L48
- FFT.transform_internal:L107
- FF...

Selected Node →

IP: http://a24e-34-124-254-60.ngrok.io

http://a24e-34-124-254-60.ngrok.io

Update

Selectd function

FFT.transform_internal:107

...aprofile/java_benchmarks/SPECjvm2008/src/spec/benchmarks/scimark/fft/FFT.java/

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.

FFT.measureFFT:calculate fft

FFT.test:performs the accuracy check on the data.

FFT.transform:compute the fast fourier transform

FFT.transform_internal:This is the actual FFT algorithm.

FFT.bitreverse:bit-revers... asm_sysvec_apic_timer_... sync_regs:NOT FOUND C...

Case-Quickly filter out unsuspecting 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.

IP:http://a24e-34-124-254-60.ngrok.io

http://a24e-34-124-254-60.ngrok.io

Update

Selectd function

FFT.transform_internal:107

...aprofile/java_benchmarks/SPECjvm2008/src/spec/benchmarks/scimark/fft/FFT.java/

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.

FFT.measureFFT:calculate fft

FFT.test:performs the accuracy check on the data.

FFT.transform:compute the fast fourier transform

FFT.transform_internal:This is the actual FFT algorithm.

FFT.bitreverse:bit-revers... asm_sysvec_apic_timer_... sync_regs:NOT FOUND C...



FF...

Case-Quickly filter out unsuspecting functions

```

102     int n = data.length/2;
103     if (n == 1) return;           // Identity operation!
104     int logn = log2(n);
105
106     /* bit reverse the input data for decimation in time algorithm */
107     bitreverse(data);
108
109     /* apply fft recursion */
110     /* this loop executed log2(N) times */
111     for (int bit = 0, dual = 1; bit < logn; bit++, dual *= 2) {
112         double w_real = 1.0;
113         double w_imag = 0.0;
114
115         double theta = 2.0 * direction * Math.PI / (2.0 * (double) dual);
116         double s = Math.sin(theta);
    
```

profile.ezview ×

SPECjvm2008-L1-cache-misses > profile.ezview

FF...transform_internal:L107 24.130401% (88774135207/36740090012)

VIRTUAL ROOT
BenchmarkThread.run:L59
BenchmarkThread.executeIteration:L82
BenchmarkThread.runLoop:L170
Main.harnessMain:L36
Main.runBenchmark:L27
FFT.main:L89
FFT.run:L246
FFT.measureFFT:L218
FFT.test:L66
FFT.transform:L48
FFT.transform_internal:L107
FF...

Quickly filter out

Focus Node

IP: http://a24e-34-124-254-60.ngrok.io

http://a24e-34-124-254-60.ngrok.io

Update

Selectd function

FFT.transform_internal:107

...aprofile/java_benchmarks/SPECjvm2008/src/spec/benchmarks/scimark/fft/FFT.java/

- 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.
- FFT.measureFFT:calculate fft
- FFT.test:performs the accuracy check on the data.
- FFT.transform:compute the fast fourier transform
- FFT.transform_internal:This is the actual FFT algorithm.
- FFT.bitreverse:bit-revers... asm_sysvec_apic_timer_... sync_regs:NOT FOUND C...

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-weight model

Try to use LLM to give auto fix suggestions

Thank you!