Efficient Implementation and Evaluation of Profilers in JavaScript-based Interpreters

Kazuki Takehi
M1 @Chiba Lab

Self-Introduction

Kazuki Takehi

Bachelor of Physics

M1 @Chiba

Poker, Hiphop, Gym

My Original Language

C-like Language

Interpreter Implemented in Javascript

Additional Feature: Profiler

```
check_def fib(n) {
  if (n < 2) {
    n;
  } else {
    fib(n-1) + fib(n-2);
  };
};

fib(10);
```

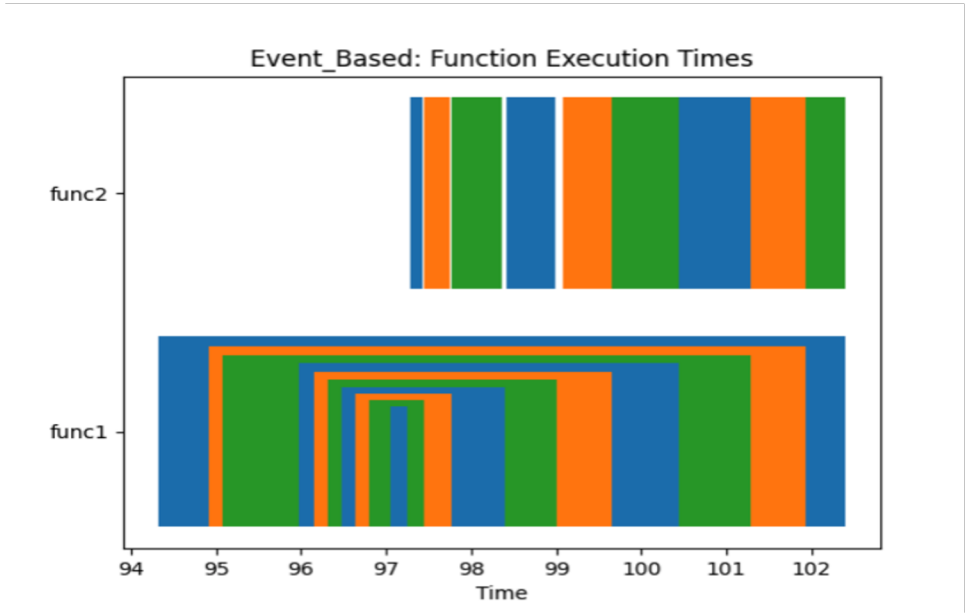
Profiler

- Tools for analyzing program execution behavior and collecting performance data. Used for debugging.

```
check_def func1(n) {
  i=0;
  while (i<100) {
    i=i+1;
  };
  if (n < 2) {
    n;
  } else {
    func1(n-1);
    func2(n-1);
  };
};

check_def func2(n) {
  i = 100*n;
  while (i > 0) {
    i=i-1;
  };
};

func1(10);
```



Event-based vs Statistical

- **Event-based Profiler**

Record stack trace at every function calls. Accurate, but high overhead

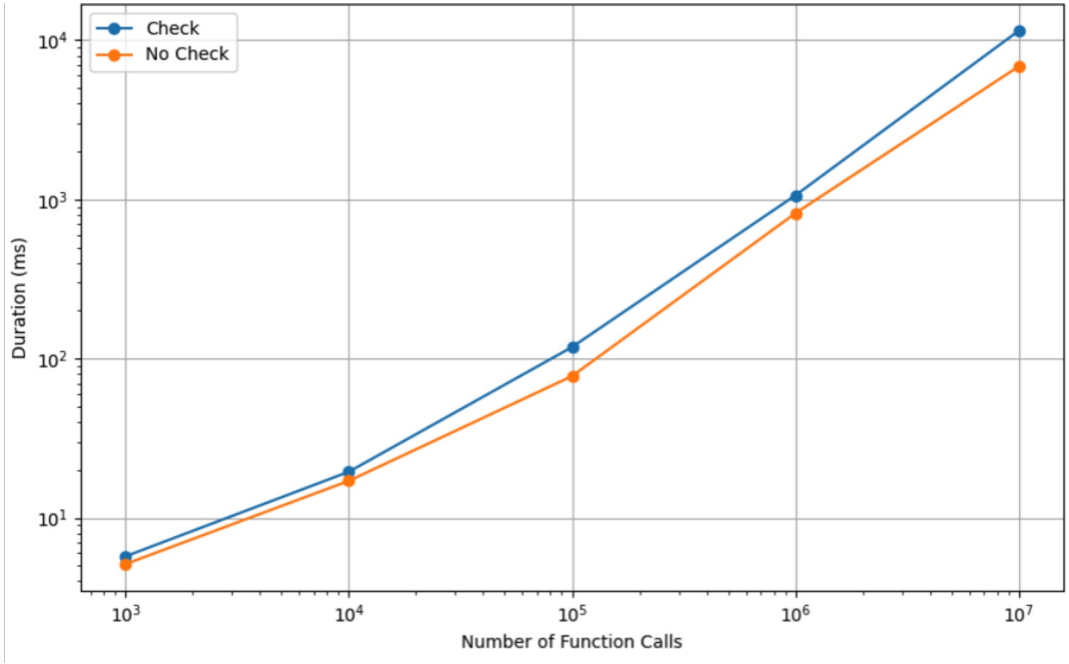
- **Statistical Profiler**

Record stack trace at regular intervals. Low overhead, less accurate

Implementing Event-based Profiler

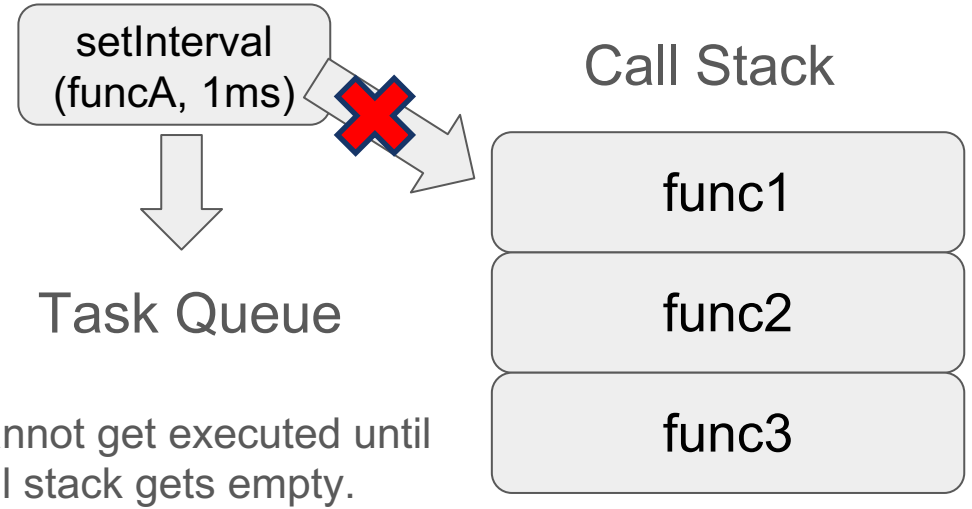
- Too slow visualization
  >10^4 function calls
- High overhead

Overhead by Event-based Profiler(Log based)

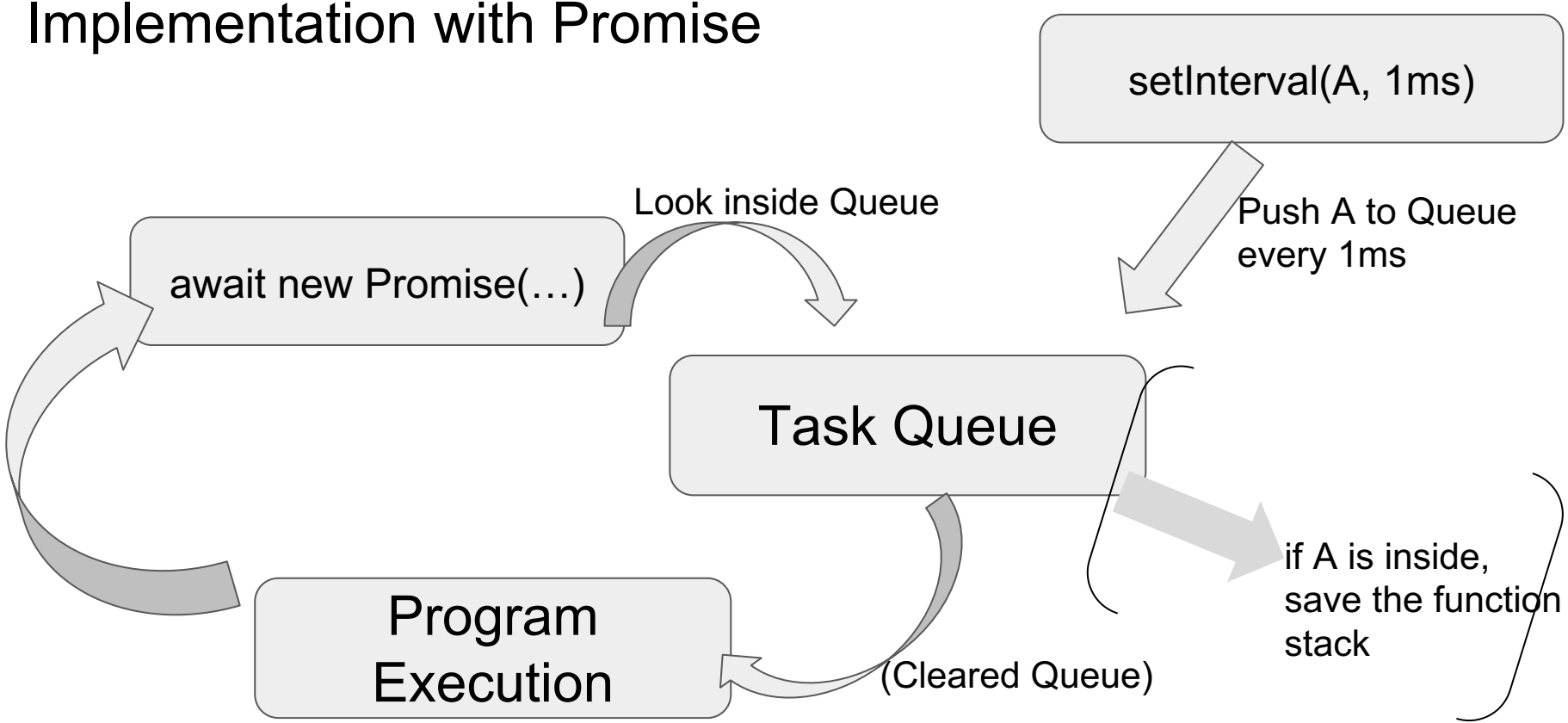


Statistical Profiling in JavaScript for my Language

- **Single Thread**
  - **Naive implementation cannot be used**
- **Approach**
  - **Promise**
  - **Worker Thread**

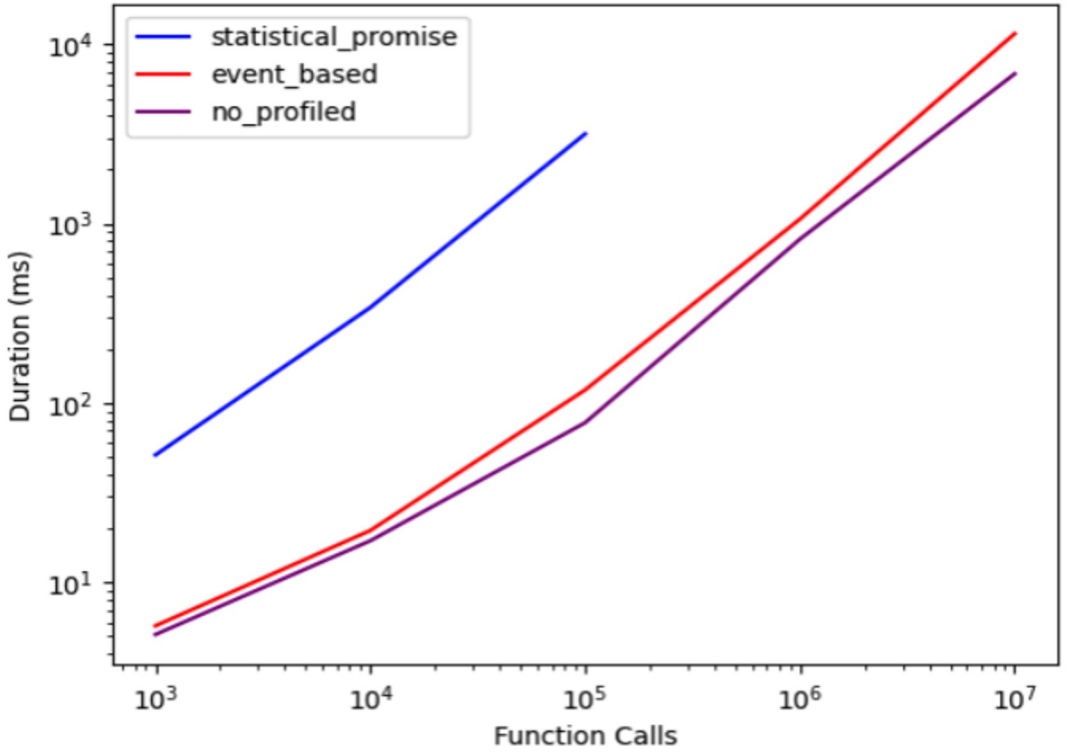


Implementation with Promise



Implementation with Promise

Making a Promise is slow

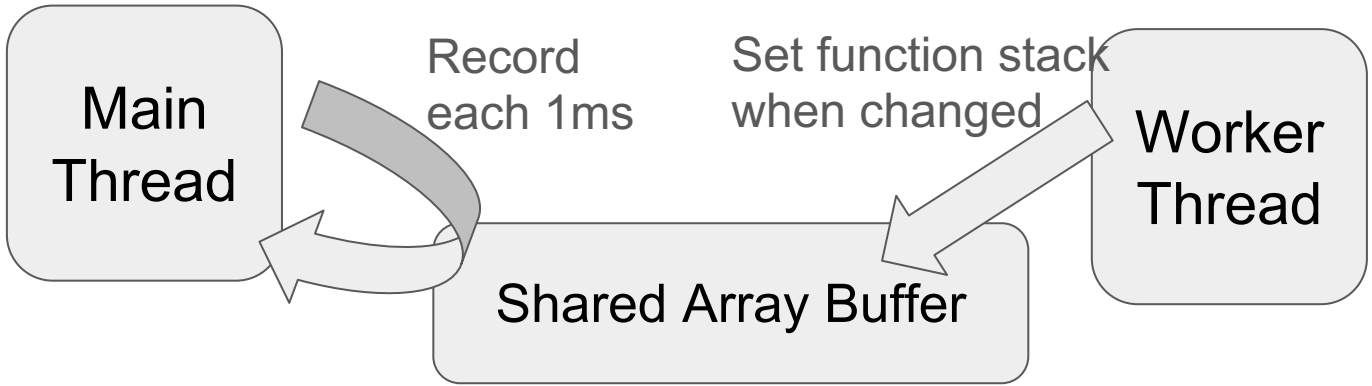


Implementation with Worker Thread

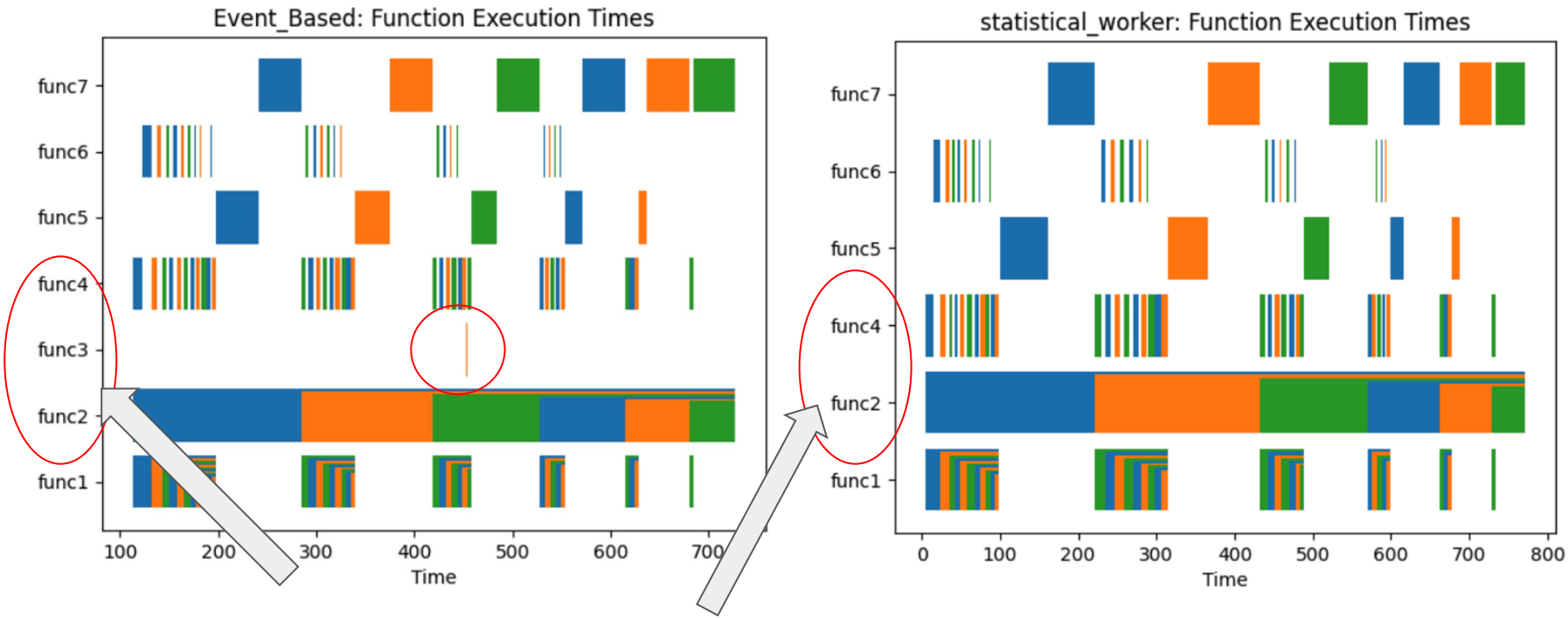
Main thread: Monitors an execution state every 1ms

Worker thread: Handles program evaluation

SharedArrayBuffer: Enables efficient communication between threads



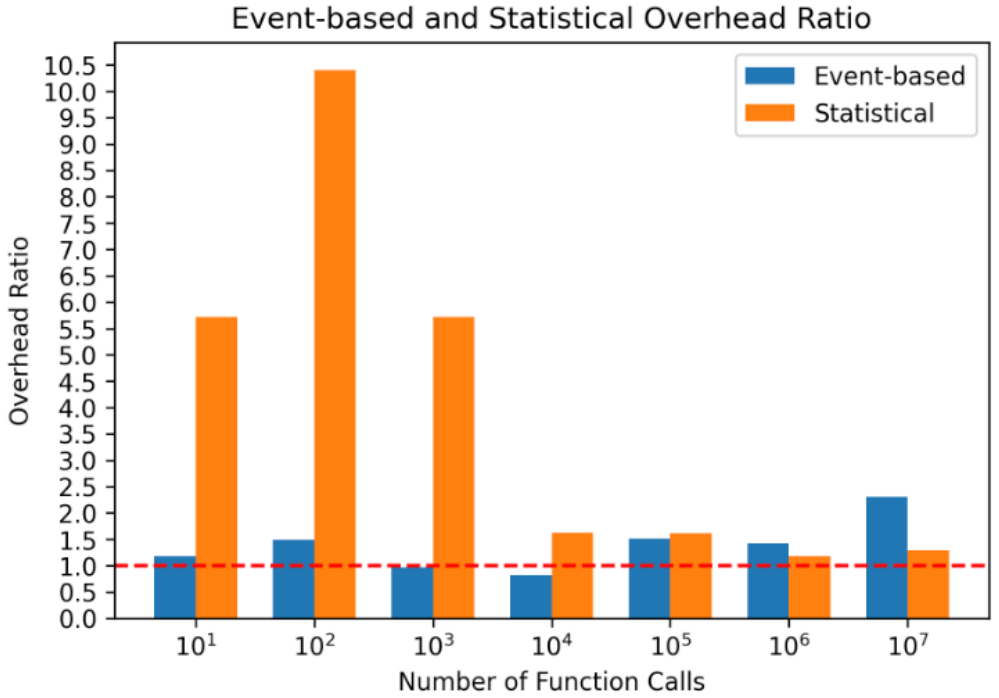
Comparison of Accuracy



func3 not recognized by Statistical Profiler

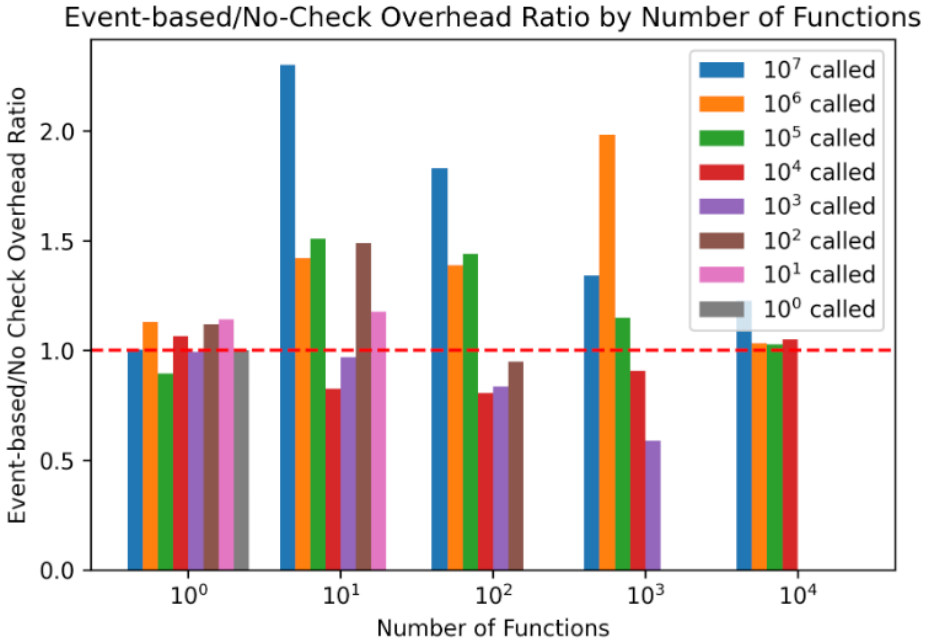
Analysis of Overhead (Part 1)

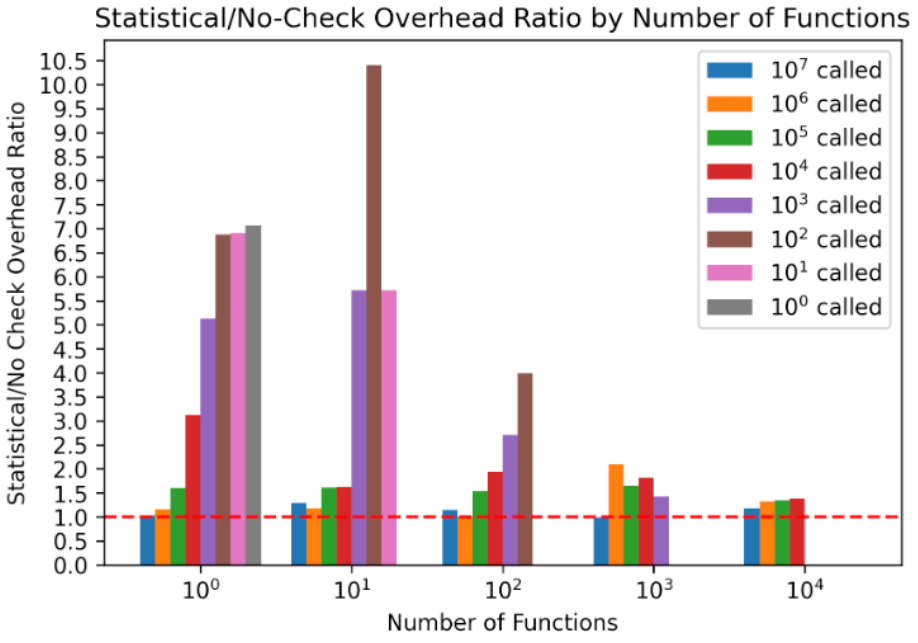
- Statistical has a large runtime overhead for $<10^3$ function calls
- Statistical becomes more efficient for $>10^5$ function calls



Analysis of Overhead (Part 2)

Number of functions did not affect much





Related Work

- Profiler for Languages with JavaScript-based Interpreters
  - Chrome DevTools
  - Dart DevTools
- Unique Features of our Language
  - Can select optimal approach based on program characteristics.
  - Implements profiler within the interpreter, enabling language-specific optimizations

Conclusion

Summary:

- Developed efficient profiling techniques for JavaScript-based interpreters.
- Developed Statistical Profiler in a Single Threaded language.
- Optimal approach: Event-based for $<10^5$ calls, Statistical for $>10^5$ calls

Future Work: Developing a profiler for a realistic language such as Python and Ruby.