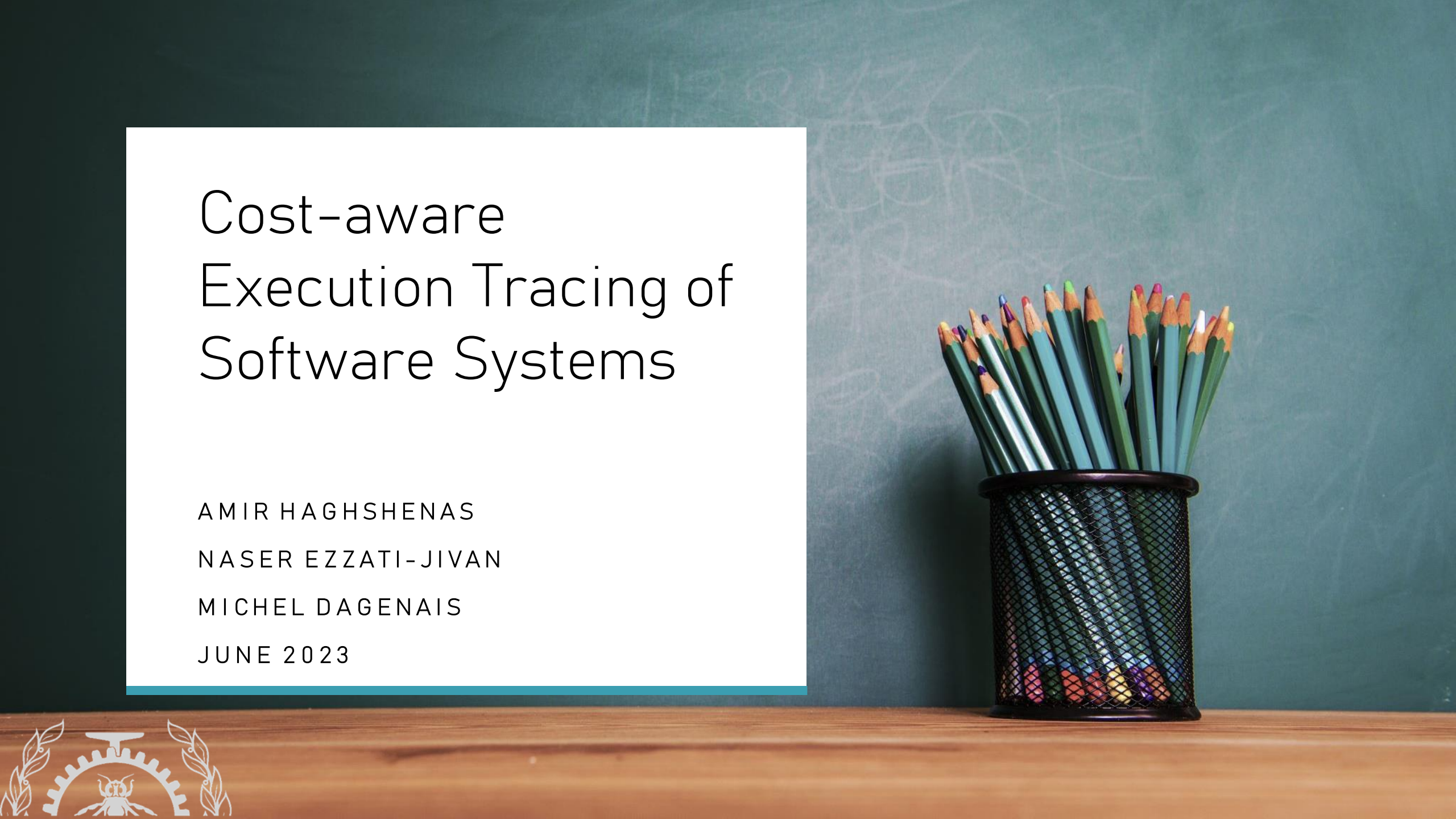
Cost-aware Execution Tracing of Software Systems

AMIR HAGHSHENAS

NASER EZZATI-JIVAN

MICHEL DAGENAIS

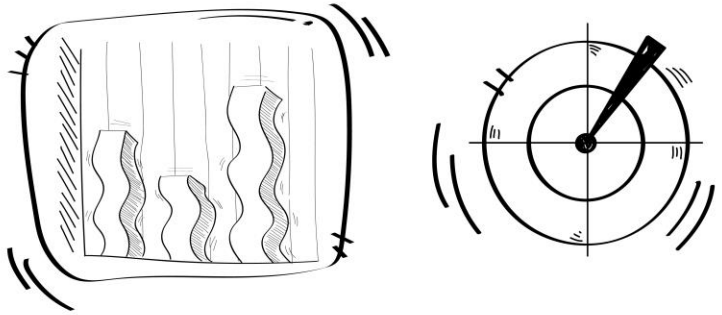
JUNE 2023



Content

- Objective
- Tracing cost definition
- Method
- Improvements
- Demo Results
- Future work

Tracing Objective



- Tracing can be done for different objectives
- Monitor system performance
  - How much time is being used by each code region.
- Not all code regions are equal in importance.
- Which code regions should we monitor?

Tracing Cost Definition

- Tracing can generate large amount of data.
- It is not always possible or desirable to trace every function.
  - Possible Large trace data
  - Limited timing capacity
- A solution is required for choosing where to trace.
  - What is the objective of tracing?
  - What is the cost of tracing?

Tracing Cost Functions

Cost Function	Definition
Execution time overhead	CPU cycles added for collecting trace data
Concurrent uniformity	the effect of tracing on concurrent behavior of a system
Memory volume overhead	the amount of memory required to collect and save tracing data
Code size	The added code size due to injection of tracepoint data
Detection delay	The delay between when a variable is defined and when it is captured
Bandwidth overhead	The overhead of transmitting the trace data over the network to processing units

Our Objective and Cost

Objective

- How unpredictable is the execution time of each code region?

Cost

- How much additional execution time will be added after tracing the application?

Method

- Using combination of static and dynamic analyses to present initial tracing configuration while respecting a minimum timing budget.
- Step 1: Calculate trace point execution time (used for cost score)
- Step 2: Calculate static metric weight factor (used for value score)
- Step 3: Use static and dynamic metrics to solve optimization problem.

Tracing Cost Calculation

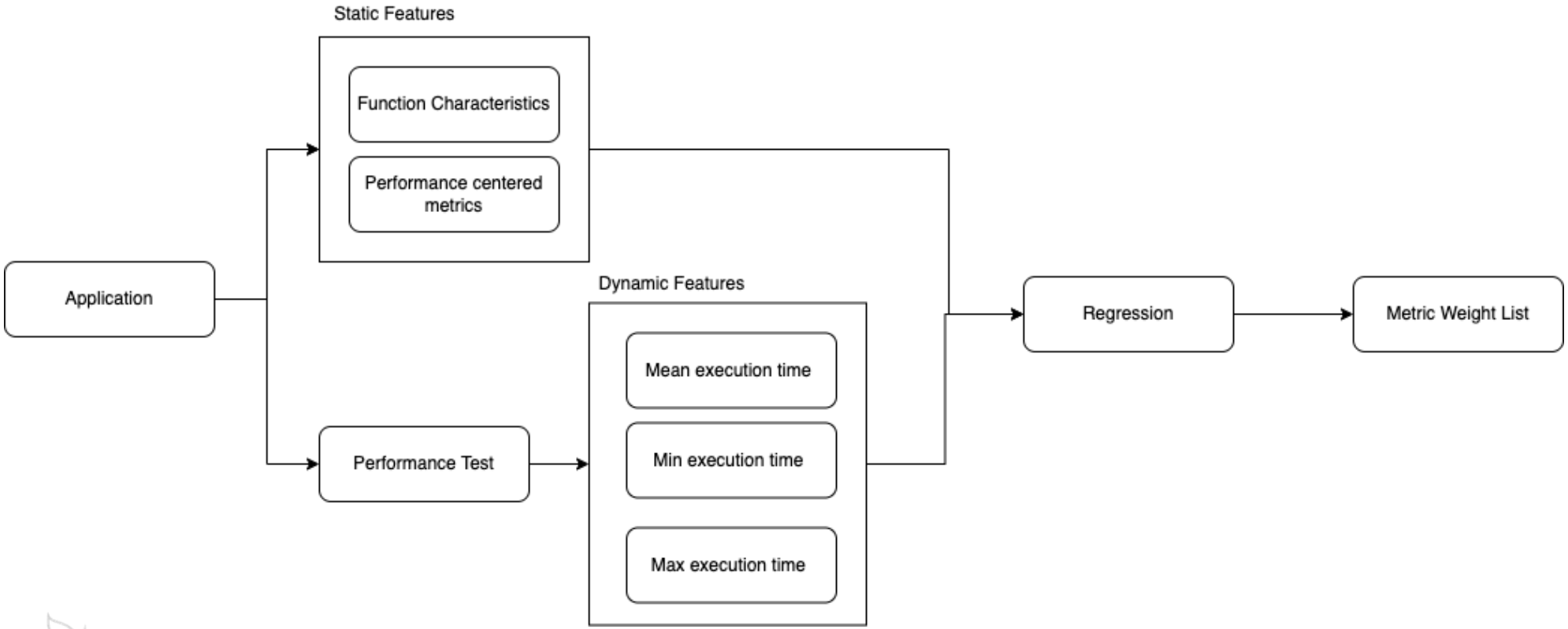
We developed a micro benchmark to calculate the execution time of calling a tracepoint

We ran the benchmark in both kernel and user spaces.

We constructed a map of execution time for different tracepoint payloads.

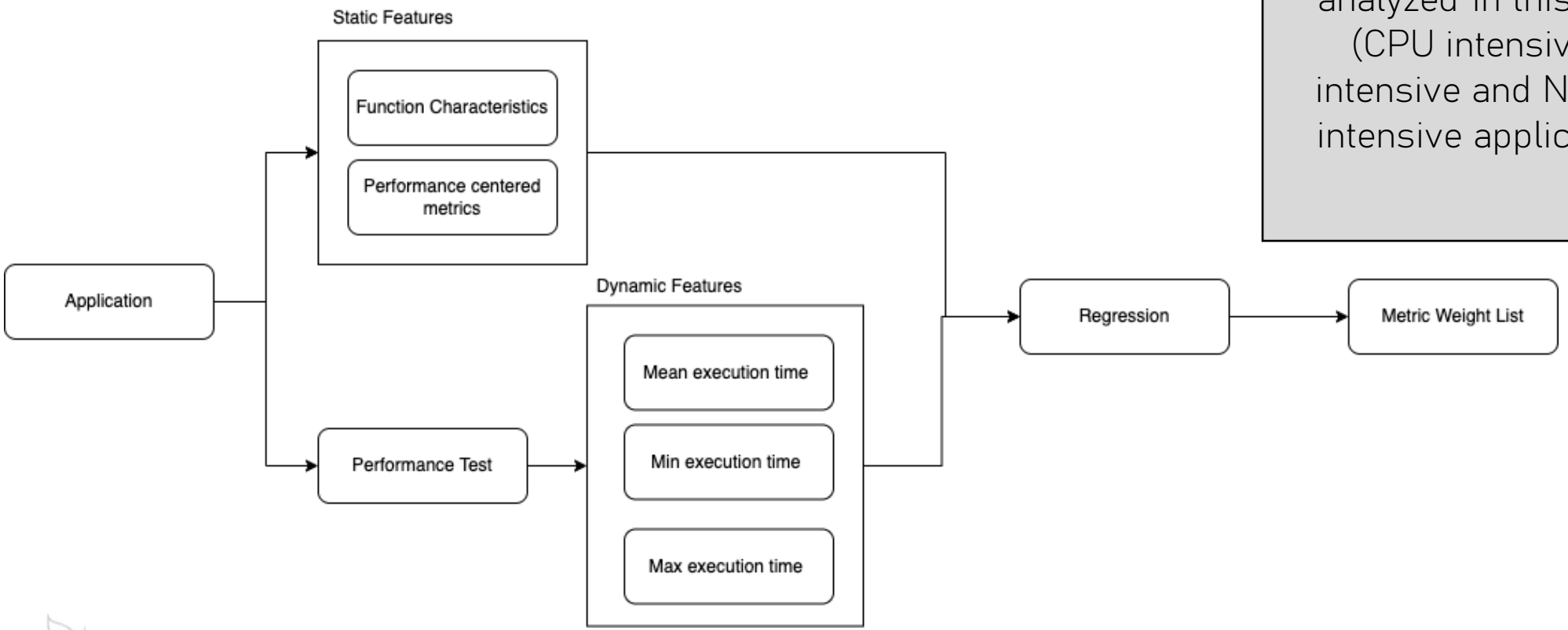
We used the mapping in predicting the tracing overhead.

Weight Factor Calculation

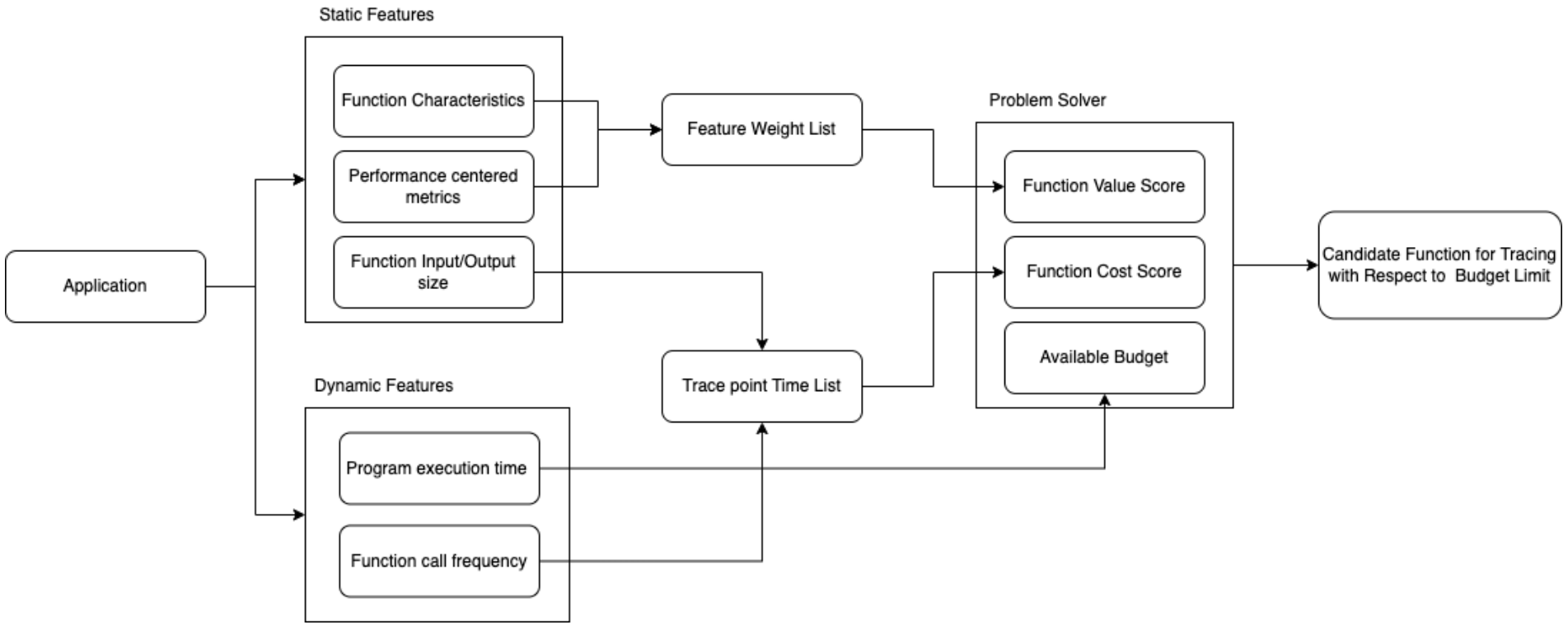


Weight Factor Calculation

Metric weight is different based on application type. Three types of application analyzed in this study (CPU intensive, IO intensive and Network intensive applications)

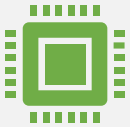


Optimization Problem



Improvements

No need for performance test for each new application (run only twice)



No additional overhead on the application run time.

Demo Results

Application domain	Data compression (CPU intensive)
Lines of code	33,000
Number of functions total	367
Duration without tracing	4m 6s
Duration with full tracing	22m 56s
Overhead allowed	5% of execution time (12s)
Number of functions selected	29

The selected functions have the most score in the probability of execution time fluctuation with respect to the provided timing budget.

Future Work

- Real-time adjustment of tracing configuration
- Using real-time phase change detection and monitoring current configuration
- Update the tracing configuration with respect to timing budget

Thank you