April 2022

# LANGUAGE AGNOSTIC CALL-GRAPH INSTRUMENTATION

Ashutosh Goel

Prabish Kumar Tattari Prakasan

# LANGUAGE AGNOSTIC CALL-GRAPH INSTRUMENTATION

AUTHORS:
Ashutosh Goel
Prabish Kumar Tattari Prakasan
Madhu

## ABSTRACT

Currently, in the space of Application Performance Monitoring (APM), instrumentation agents for different languages capture stack-traces, as well as additional information indicating how functions are called, how these calls are related to each other, and how much time each function call takes, which can be represented using call-graphs. However, the underlying software (code) used to capture these call-graphs is often implemented in different programming language-specific agents (e.g., a Java-agent, a php-agent, a Python-agent, etc.), which results in code-logic-duplication and maintenance overhead. It would be beneficial to provide a capability for capturing call-graphs in a language agnostic manner. Techniques presented herein provide for the ability to implement an agent through the use of Extended Berkley Packet Filter (eBPF) technology such that the agent is not tied to any specific language and can capture call-graphs and other context-metadata from many different applications, which may improve observability across multiple languages.

## DETAILED DESCRIPTION

Application Performance Monitoring (APM) agents as currently implemented in application environments typically collect telemetry data from applications and also collect call-graphs. These call-graphs typically contain information indicating how functions/methods in an application are called and how they are related (i.e., which function called which other function, how much time was taken by each function to execute and other metadata related to the hierarchy of these functions). Additionally, some APM agents sample stack-traces at a regular frequency of a running application such that the stack-trace can be attached and reported in the context of a captured call-graph.

This proposal provides a technique through which a language-agnostic agent may be implemented, such that the agent is not tied to any specific language, yet, can still capture call-graphs and other context-metadata from applications for observability.

In particular, the solution presented herein utilizes Extended Berkley Packet Filter (eBPF) technology through which a standalone eBPF agent can be built that facilitates capturing call-graphs of applications that may be written in different programming languages. This eBPF agent can intercept every method/function call in an application using user-probes.

These interceptions can be used to capture information of different functions and/or methods, including, but not limited to, function-name, line-number from where a function was called, the file-name from where the call was made, etc. and the captured information can be stored into shared-memory data-structures (i.e., memory that is shared between the user-space and the kernel-space) called eBPF-Maps, as typically known in the technology space. The information stored in the eBPF-maps can be collected in user-space and used to generate call-graphs.

In addition, the collected call-graph can be correlated with data from telemetry libraries like open-telemetry. To achieve this, when the eBPF agent is intercepting methods/functions used in an application mentioned above, it can also look for function calls from open-telemetry library (telemetry library) for starting and ending spans to capture span-ids, trace-ids, and other metadata that may be relevant to the context of a span/trace. The previously collected call-graph information can be stitched together with a corresponding span-context for correlation, which can be used by an observability backend. Figure 1, below, illustrates an example operational flow associated with such techniques involving an eBPF agent, as proposed herein.

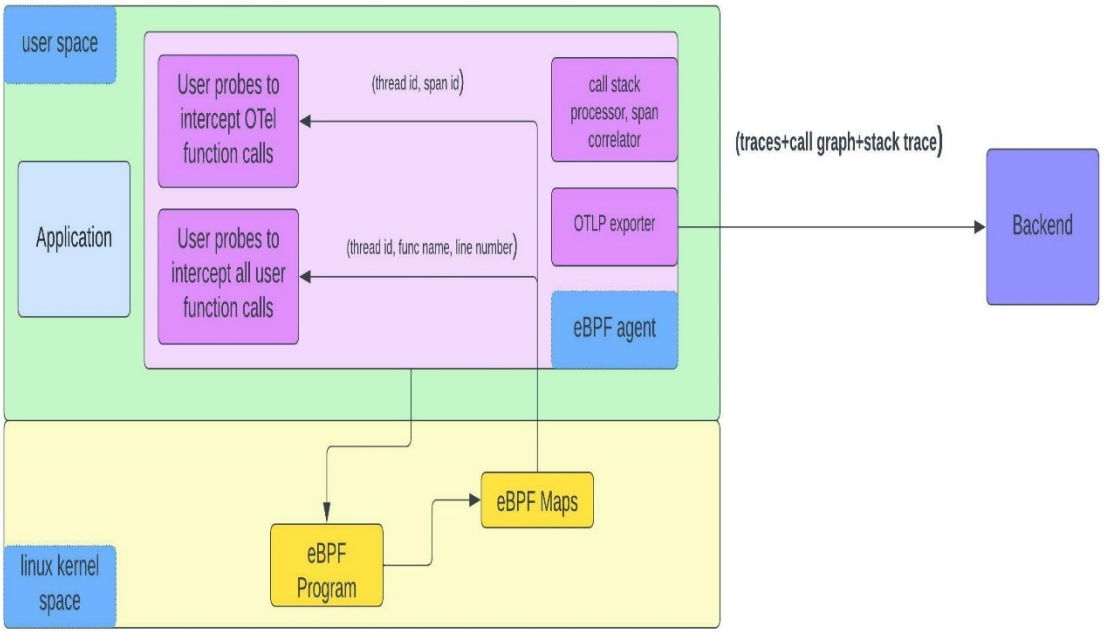


*Figure 1: Example Operational Flow*

Thus, this proposal facilitates implementation of a language agnostic agent that can be built using eBPF that may provide the ability to capture call-graphs for different applications. Conventional eBPF technology does not need to be modified or extended in order to achieve such an agent. Rather, conventional eBPF technology can be leveraged in order to generate call-graphs in a language agnostic manner.

Currently, agents are often developed using language-specific constructs (e.g., Java, nodejs, php, etc.) in order to capture call-graphs, yet, this proposal provides a solution that leverages eBPF facilitate call-graph capturing in language agnostic manner. Although techniques herein are described with reference to instrumentation agents, it is to be understood that such techniques could also be extended to facilitate language-agnostic profiling agents, performance agents, or the like.

Accordingly, techniques presented herein may advantageously provide a solution through which a single language agnostic agent, driven by eBPF, can be used to capture call-graphs and other instrumentation data, rather than using language-specific agents. Further, since the interception logic of an eBPF agent operates in kernel-space, the solution provided herein is likely to be more performant in comparison to regular user-space agents that are often utilized to collect call-graphs.