Typical sampling profilers periodically interrupt the observed application and record a stack trace of its threads, i.e., the executing methods and callers. The entirety of the collected samples gives a statistical picture of where the application spends its time. However, interrupting application threads and delaying their execution while building a stack trace can cause a significant performance impact, especially in common Java VMs, which require all application threads to be paused together.

Our research group has devised a fast alternative approach for sampling in a Java VM [1]. This approach uses the perf subsystem of the Linux kernel [2] to periodically trigger interrupts with a hardware timer. The interrupt handler of perf only copies a memory fragment from the stack of the currently executing thread to a buffer and then resumes execution. The stack fragments are asynchronously read from the buffer and decoded to Java stack traces. This approach is very lightweight because the interrupts and copy operations are inexpensive and decoding can be done with minimal impact on the application. However, the disadvantage of our implementation is that it requires specific operating system capabilities.

The task of this thesis is to adapt our approach to use a more portable mechanism, such as POSIX signals [3]. This entails periodically interrupting threads in software, copying a stack fragment to a buffer when interrupted, and ensuring safe concurrent access to the buffer(s). The performance of the implementation should be compared to that of our existing approach.

Dipl.-Ing. Peter Hofer