Online OS Profiling with SnailTrail

Distributed Systems Lab Project
(can be extended to a Master Thesis)

Understanding the performance of multi-threaded systems is hard. In case the target system is a complex OS running thousands of applications, then the problem becomes even more challenging.

Traditional profilers like `gprof` collect time statistics for the running programs (essentially the time spent per function call), which can then be aggregated to estimate how much time was spent in each function during the program execution. If the program is executed sequentially, i.e. by a single thread, then this naïve approach can also be used to estimate the contribution of a function call to the overall latency of the execution. In concurrent programs, however, *causality* matters: different parts of the program may be executed by different threads whose activities may overlap in time, thus, the simple summation of individual times is incorrect.

Critical Path Analysis (CPA) tackles this problem by introducing the notion *dependencies* between activities (e.g. function calls) in a program execution. Intuitively, an activity X depends on another activity Y iff Y must first finish before executing X. Activities can be represented as edges in a graph where the nodes denote start and end events, each one associated with a timestamp. This abstract graph representation is known as the Program Activity Graph (PAG). The critical path is then defined as the longest path in the PAG, i.e. the sequence of dependent activities whose duration equals the total duration of the program execution.

Existing CPA approaches are defined on the complete PAG that corresponds to a finished program execution. As a result, they cannot be applied in a scenario where the instrumented system is continuously running like in the case of an OS. In such a setting, traditional CPA needs significant modifications and can be performed only within time windows. To this end, we have recently designed and implemented SnailTrail [3], a novel system that introduces the notion of *transient critical paths* for online CPA. Online CPA has already been applied to popular streaming data processing systems with promising results.

The goal of this project is to take SnailTrail one step further and perform online CPA of a modern operating system. Our use case will be Barrelfish [4], a research OS designed and implemented in the Systems Group at ETH Zurich. Barrelfish is heavily instrumented and is an ideal testbed not only for gaining insights into the dependencies of low-level OS tasks, but also for benchmarking the performance of SnailTrail itself.

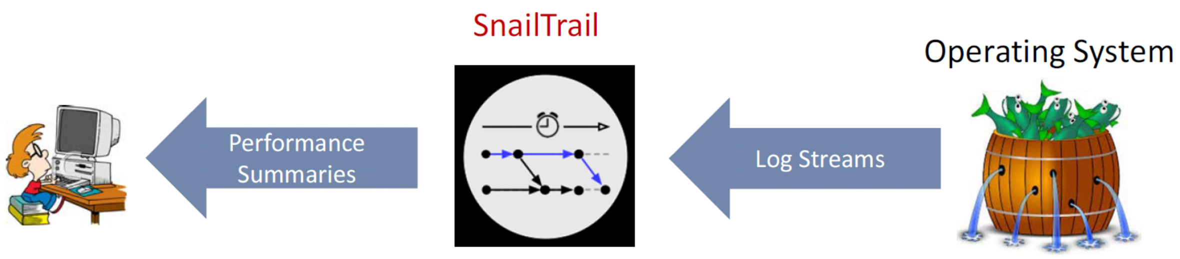
As a DSL project, the core part of the work lies in the integration of Barrelfish with SnailTrail—the latter currently accepts a different log format. However, the project can also be extended into a Master Thesis that will investigate possible improvements of SnailTrail with ideas from non-intrusive approaches, e.g. [1], and other related works in the field, such as [2].

[1] Charlie Curtsinger and Emery D. Berger. “Coz: Finding Code That Counts with Causal Profiling”. In: *Proceedings of the 25th Symposium on Operating Systems Principles*. SOSP ’15. Monterey, California: ACM, 2015, pp. 184–197.

[2] Nikolai Joukov et al. “Operating System Profiling via Latency Analysis”. In: *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7*. OSDI ’06. Seattle, WA, 2006, pp. 7–7.

[3] Ralf Sager et al. “SnailTrail: Online Bottleneck Detection for your Dataflow”. In: *Proceedings of EuroSys*. 2017.

[4] *The Barrelfish Operating System*. URL: `http://www.barrelfish.org/`.



Interested?

**Please contact John Liagouris (liagos@inf.ethz.ch) and Moritz Hoffmann (moritz.hoffmann@inf.ethz.ch).
The proposed project will be supervised by Dr. John Liagouris and Prof. Timothy Roscoe.**