# SnailTrail: Online Bottleneck Detection for your Dataflow

†Ralf Sager, *John Liagouris, Desislava Dimitrova, ‡Andrea Lattuada, Vasiliki Kalavri
‡Zaheer Chothia, ‡Moritz Hoffmann, Timothy Roscoe

Systems Group, Department of Computer Science, ETH Zürich
firstname.lastname@inf.ethz.ch, †*MSc student*, ‡*PhD student*, **poster presenter*

***Motivation*** Understanding the performance of large-scale data processing applications is hard. In distributed dataflow systems, the computation of several parallel processes is interleaved with data and control communication and execution dependencies typically span multiple system components. In such an environment, bottleneck detection is cumbersome and currently relies heavily on humans. After decades of systems research, the state-of-the-art in performance analysis still relies on offline trace processing, thus it is only suitable for batch computations and post-mortem reports. This work presents `SnailTrail`, a novel framework for *online* performance analysis in modern dataflow engines that can identify bottlenecks in real-time and make automated optimization possible at runtime.

***Our Approach*** We make the observation that modern dataflow systems are built on top of common execution primitives. Based on that, we introduce a general instrumentation methodology that enables tracking of *important* events in the execution of a dataflow with negligible performance overhead. `SnailTrail` uses the generated event streams to construct and continuously maintain an evolving graph model of system activities, inspired by the concept of Program Activity Graphs (PAGs) in critical path analysis [1]. In contrast to existing approaches, our framework provides *online performance analytics at scale*: execution bottlenecks are identified at runtime and within configurable fine-grained time windows. This online analysis provides unprecedented performance insights on long-running computations (e.g. in deep learning) and continuous computations on unbounded data (e.g. in IoT applications) where traditional critical path analysis is not applicable.

`SnalTrail` relies on the notion of *transient critical paths*, a time-oriented adaptation of the standard critical path. Transient critical paths serve as a "signature" of the dataflow execution, encoding valuable performance metrics, while also possessing a set of interesting properties that can serve as lightweight rules to verify the correctness of the instrumentation itself. Interestingly, transient critical paths are also resilient to clock skewness and incomplete activity logs (which naturally occur in distributed system tracing), and their computation can be efficiently parallelized.

***Early Results*** We have implemented the core engine of `SnailTrail` on top of Timely Dataflow [3], a general-purpose streaming system with native support for data-parallel computations. `SnailTrail` is designed to analyze the performance of dataflow systems with hundreds, even thousands, of parallel workers in near-real time and with modest computational resources. As a proof of concept, we have applied our methodology to Timely Dataflow itself. We show how the transient critical paths can be used to generate online performance summaries, which are more informative than summaries provided by offline analyzers [2, 4].

***Ongoing Work*** We are currently extending `SnailTrail` to support more dataflow systems, namely Apache Spark, Apache Flink, and TensorFlow. Our goal is to reach beyond real-time performance summaries and enable applications such as straggler mitigation, performance regression detection across different software versions, and online what-if analysis. Further, we plan to leverage transient critical paths to support automatic performance optimization at runtime, e.g., via adaptive resource allocation and scheduling strategies. This direction is in line with our broader vision for a next generation of self-tuned dataflow systems.

## References

[1] BÖHME, D., ET AL. Scalable critical-path based performance analysis. In *IPDPS* (2012).

[2] CHOW, M., ET AL. The mystery machine: End-to-end performance analysis of large-scale internet services. In *OSDI* (2014).

[3] MURRAY, D. G., ET AL. Naiad: A timely dataflow system. In *SOSP* (2013).

[4] OUSTERHOUT, K., ET AL. Making sense of performance in data analytics frameworks. In *NSDI* (2015).

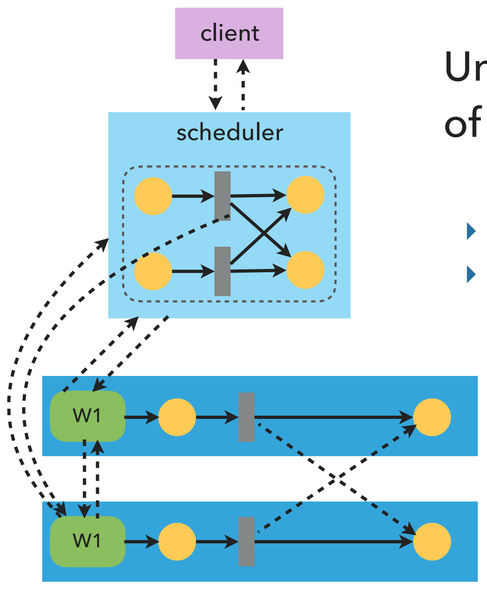

*EuroSys'17*,

http://dx.doi.org/10.1145/.

# SnailTrail: Online Bottleneck Detection for your Dataflow

Ralf Sager, John Liagouris, Desislava Dimitrova, Andrea Lattuada,
Vasiliki Kalavri, Zaheer Chothia, Moritz Hoffmann, Timothy Roscoe

## WHY IS MY DISTRIBUTED PROGRAM SLOW?

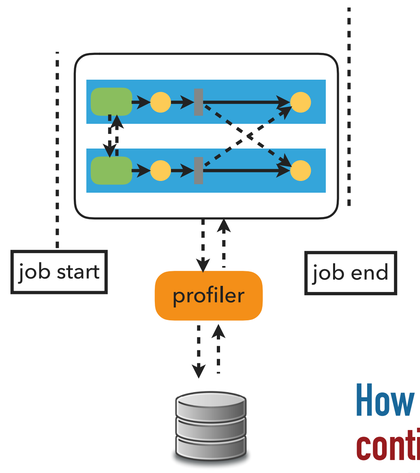



Understanding the performance of distributed data processing

- many processes and activities
- computation is interleaved with data and control communication
  - execution dependencies are not easy to infer and might be dynamic
  - the cause of a bottleneck is usually not isolated but is a chain of events spanning multiple processes

## POST-MORTEM ANALYSIS IS EASY

1. Collect traces during execution

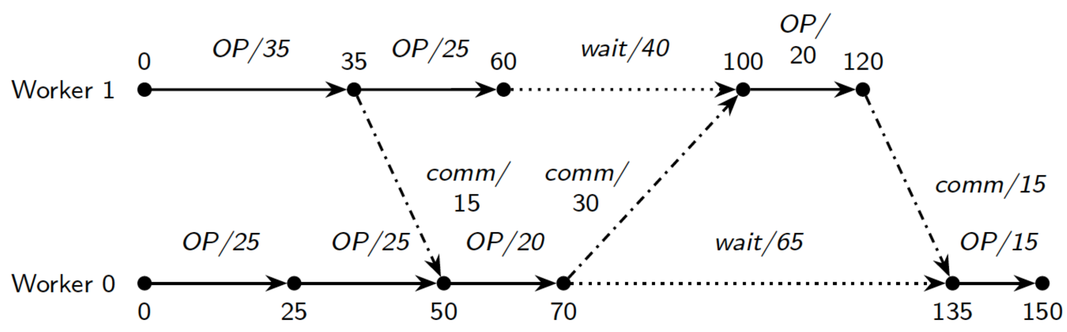



2. Analyze traces *offline*


- Critical path analysis
- Performance summaries

**How to compute the critical path for continuously running, dynamic distributed applications, with unbounded input?**

## THE EVOLVING PROGRAM ACTIVITY GRAPH

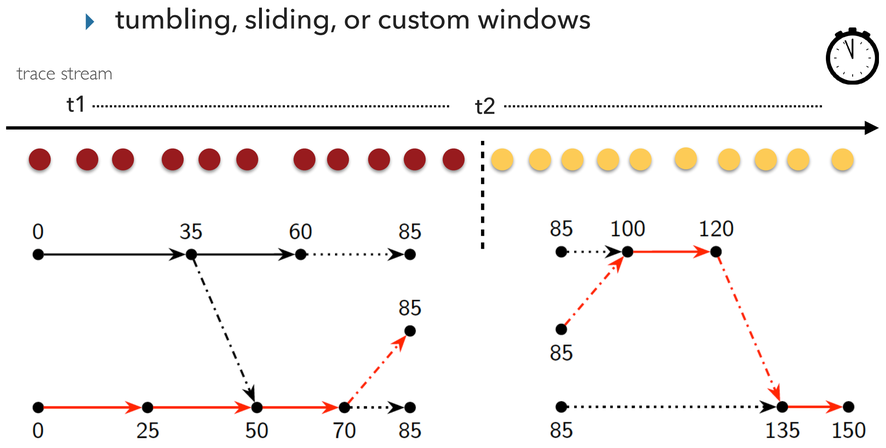



An *evolving* time-annotated graph model that captures computation and communication dependencies among distributed workers

- Vertices represent the start or end of activities and communication
- Edges represent the duration of activities and communication

## TRANSIENT CRITICAL PATH ANALYSIS

- Continuous computation of multiple *transient* critical paths on trace *snapshots*
  - tumbling, sliding, or custom windows



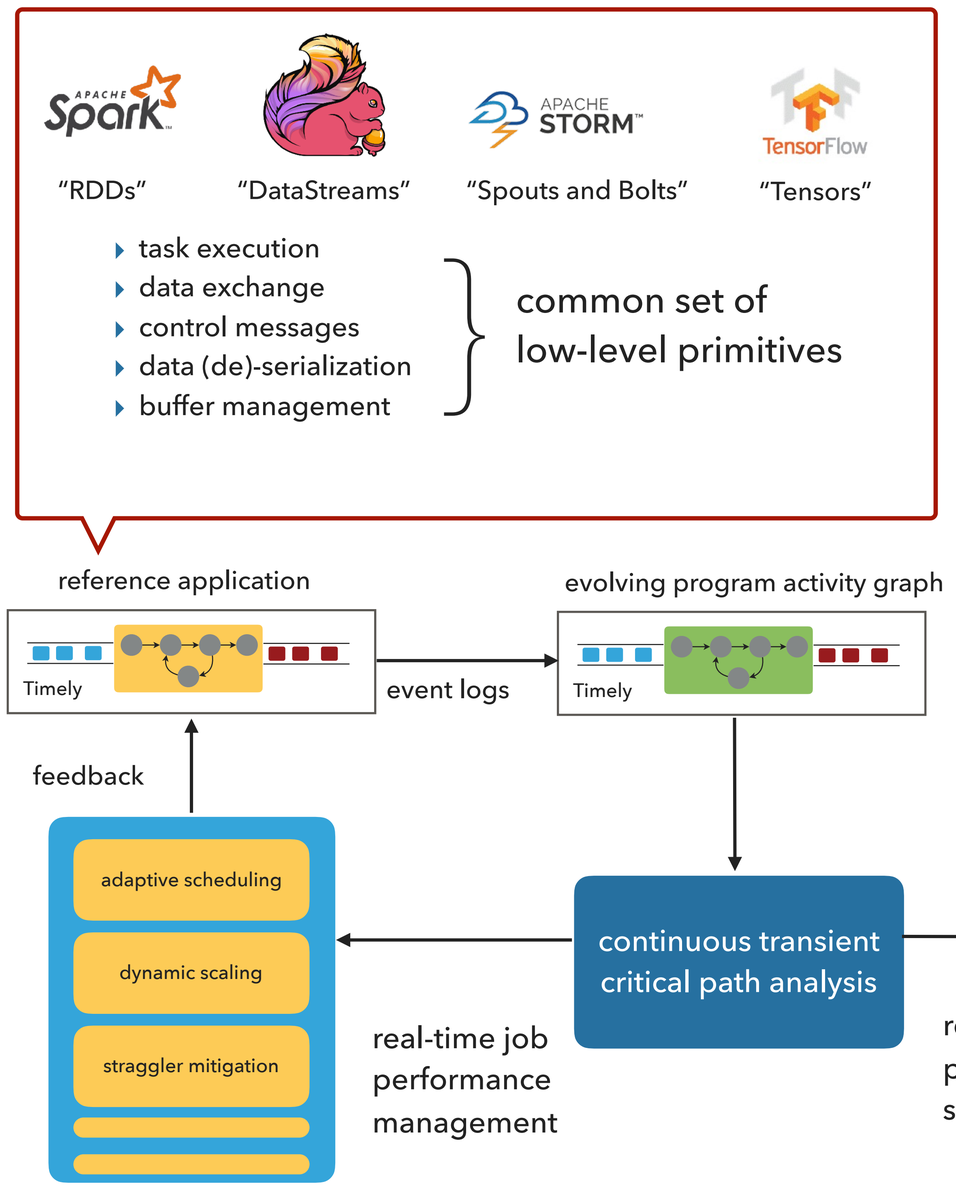

- aggregate analysis over multiple transient critical paths
- online betweenness centrality to find the most "central" activities
- online graph pattern matching to detect potential bottlenecks

## SYSTEM ARCHITECTURE

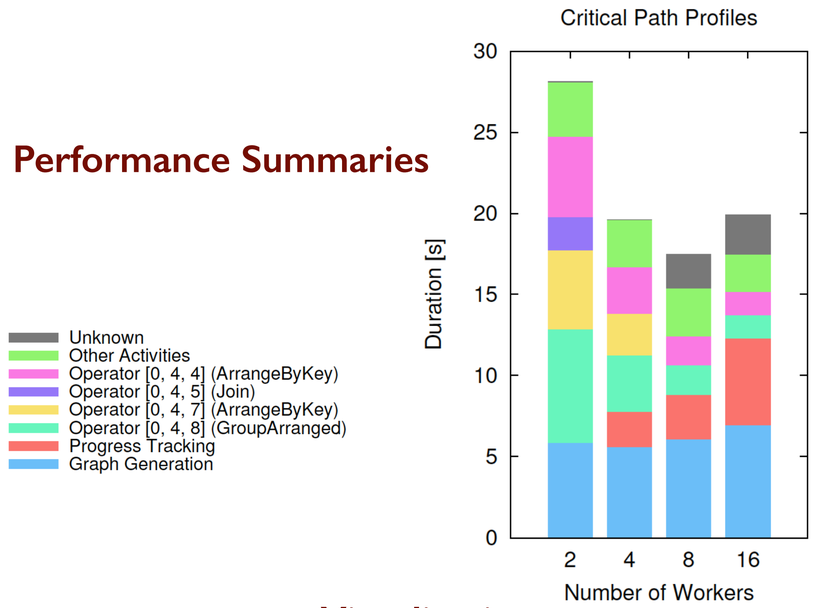
"RDDs" "DataStreams" "Spouts and Bolts" "Tensors"

- task execution
- data exchange
- control messages
- data (de)-serialization
- buffer management

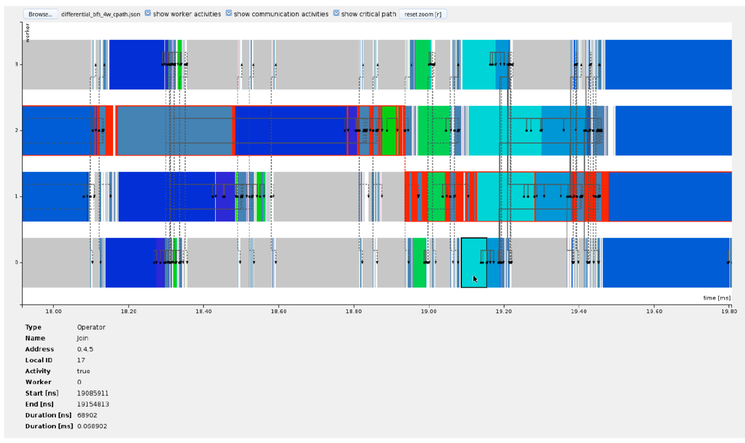
} common set of low-level primitives




### Performance Summaries




### Visualization




Systems Group, Department of Computer Science, ETH Zurich
firstname.lastname@inf.ethz.ch