Online performance tuning of distributed streaming dataflows with SnailTrail

Master Thesis Proposal

Introduction

To better understand the performance of distributed dataflow computations, we have built SnailTrail [1, 2], a tool for online performance analysis and bottleneck detection in distributed streaming dataflows. SnailTrail performs critical path analysis on live execution traces and analyzes a graph model of the execution state called the the Program Activity Graph (PAG). In the PAG, an edge represents the duration of a worker or communication activity, such as processing, serialization, sending a data message, etc. Currently, SnailTrail computes a betweenness centrality-based metric on snapshots of execution traces and ranks worker activities based on their potential for optimization benefit. The output is a set of continuously updating performance summaries that potentially reveal critical activity types, skew, stragglers, and communication bottlenecks.

Thesis Goal

This thesis aims at building a framework for online performance optimization of distributed streaming applications based on SnailTrail summaries. In particular, the student will choose one or more reference systems (Timely, Apache Flink, Spark, Heron, Tensorflow) and define a set of applications and performance evaluation scenarios to be implemented and tested. For instance, we can focus on straggler mitigation or critical communication among parallel workers. To reveal interesting structural and temporal dependencies in the PAG, the student will extend SnailTrail's performance summaries with centrality and ranking graph algorithms, such as random-walk betweenness centrality [3], random-walk closeness centrality [4], and load centrality [5]. Finally, the student will evaluate the effectiveness of performance summaries using the proposed metrics and applications.

Evaluation and Measurements

The evaluation work of this thesis consists of two parts. First, the student will evaluate the implemented framework in terms of performance. As an online profiler, SnailTrail is required to operate with extremely low latency in order to enable real-time performance tuning. At the same time, SnailTrail's graph analytics must be scalable, so that the tools can comfortably analyze traces from large deployments with hundreds of parallel workers. The second evaluation part considers the effectiveness of performance summaries. Using Timely, Apache Spark, Flink, or TensorFlow, the student will define use-cases and evaluate optimization potential and bottleneck detection capabilities of available PAG metrics. We will investigate whether the metrics reveal causal dependencies among worker activities or other interesting properties of the execution. The goal of this evaluation will be to confirm which of the performance summaries can be useful in practical scenarios to tune the performance of distributed dataflow applications.

Additional Directions

The student can additionally explore how graph-structured models of execution dependencies could be leveraged in what-if simulations. In particular, we will investigate how the PAG can be used to predict the performance effects of optimization actions. We will define a set of transformations on graph models of execution traces and express SnailTrail-driven optimizations as a series of such transformations with predictable performance benefits.

References

[1]: https://github.com/strymon-system/snailtrail
[2]: https://www.usenix.org/system/files/conference/nsdi18/nsdi18-hoffmann.pdf
[3]: M.E. J. Newman, A measure of betweenness centrality based on random walks, Social Networks, Volume 27, Issue 1, 2005.
[4]: en.wikipedia.org/wiki/Random_walk_closeness_centrality
[5]: Ulrik Brandes, On variants of shortest-path betweenness centrality and their generic computation, Social Networks, Volume 30, Issue 2, 200

If you are interested in this project please contact Vasiliki Kalavri (kalavriv@inf.ethz.ch). The proposed thesis will be supervised by Prof. Timothy Roscoe.