Rethinking how distributed applications are built

Till Rohrmann
Apache Software Foundation
trohrmann@apache.org

ABSTRACT

In our more and more connected world where people are used to managing their lives via digital services, it has become mandatory for a successful company to build applications that can scale with the popularity of the company's services. Scalability is not the only requirement but similarly important is that modern applications are highly available and fast because users are not willing to wait in our ever faster moving world. Due to this, we have seen a shift from the classic monolith towards micro service architectures which promise to be more easily scalable. The emergence of serverless functions further strengthened this trend more recently. By implementing a micro service architecture, application developers are all of a sudden exposed to the realm of distributed applications with its seemingly limitless scalability but also its pitfalls nobody tells you about upfront. So instead of solving business domain problems, developers find themselves fighting with race conditions, distributed failures, inconsistencies and in general a drastically increased complexity. In order to solve some of these problems, people introduce endless retries, timeouts, sagas and distributed transactions. These band aids can quickly result in a not so scalable system that is brittle and hard to maintain. The underlying problem is that developers are responsible for ensuring reliable communication and consistent state changes. Having a system that takes care of these aspects could drastically reduce the complexity of developing scalable distributed applications. By inverting the traditional control-flow from application-to-database to database-to-application, we can put the database in charge of ensuring reliable communication and consistent state changes and, thus, freeing the developer to think about it. In this keynote, I want to explore the idea of putting the database in charge of driving the application logic using the example of Stateful Functions, a library built on top of Apache Flink that follows this idea. I will explain how Stateful Functions achieves scalability and consistency but also what its limitations are. Based on these results, I would like to sketch the requirements for a runtime that can truly realise the full potential of Stateful Functions and discuss with you ideas how it could be implemented.

ACM Reference Format:
Till Rohrmann. 2022. Rethinking how distributed applications are built. In *The 16th ACM International Conference on Distributed and Event-based Systems (DEBS '22), June 27–30, 2022, Copenhagen, Denmark.* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/3524860.3544410

DEBS '22, June 27–30, 2022, Copenhagen, Denmark

ACM ISBN 978-1-4503-9308-9/22/06.
https://doi.org/10.1145/3524860.3544410

1 BIOGRAPHY

Till is a PMC member of Apache Flink and a cofounder of Ververica. During his time at Ververica, his work focused on enhancing Flink's scalability and high availability. He also bootstrapped Flink's CEP and the initial ML library. Nowadays, Till focuses on making the development of distributed applications easier by employing stream processing techniques to this space.