Themelios: a model-checked reimplementation of Kubernetes

Andrew Jeffery, Richard Mortier
University of Cambridge

Orchestration platforms are critical for today's service deployments improving reliability, automation and delivery. Kubernetes is the most prevalent such platform, being particularly widely used in the cloud-native domain. However, due to its popularity Kubernetes is now being pushed towards the edge where the deployment environment has very different characteristics.

Deployment of Kubernetes at the edge places services close to users but does so using an architecture that doesn't properly accommodate differences in latency, reliability, and connectivity. Spreading a single instance of Kubernetes across multiple edge sites leads to availability problems, limiting each site's ability to perform operations in the face of dynamic requirements. Custom controllers for Kubernetes have been proposed that seek to address some of these challenges, but they retain the same core architecture and so do not address the core problems. The main distributions for Kubernetes at the edge encompassing these custom controllers are KubeEdge [1] and K3s [2], which either still rely on the cloud or are not designed to be distributed across multiple edge sites. There is currently no solution to the problem of how to run a single cluster across multiple sites that both provides ease of use and retains reliability, availability, and local-first operation.

The underlying limitation that prevents realising this architecture is the central key-value store used to replicate state — in Kubernetes, etcd. Its consistency requirements mean that it does not support local operation for every edge site. Thus, a natural approach to rearchitecting Kubernetes to support edge-site deployments is to weaken these consistency requirements. Unfortunately, reasoning about how changes to such a core component will affect Kubernetes behaviour is difficult as there is no clear statement of the properties Kubernetes requires from its key-value store, and therefore no straightforward way to check whether they are provided by any alternative store.

In Themelios we consider this problem of ensuring that different architectures for orchestration platforms suiting different environments provide the required behaviours. We start by developing an understanding of what *correctness* is in Kubernetes specifically as the basis for orchestration more generally. Our poster presents how and why we have reimplemented the core of Kubernetes, and used model checking to begin to extract guarantees being made in preparation for testing them against new architectures. A key feature of our approach is to maintain the deployable nature of the components that we check, for correctness and trust. This means that components we check can integrate into an existing Kubernetes cluster, gaining the benefits of our checked approach, or can themselves be used to build a new cluster.

Having checked components against properties extracted from the core components, we then seek to address the architectural changes required to make Kubernetes more edge-suitable. The primary architectural change we focus on is consistency at the core of the cluster, used to store the desired state and currently observed state of resources. This change in consistency would enable the core to be rearchitected to match the deployment environment. Notably, issues already exist for Kubernetes relating to staleness of read values, leading to a critical safety problem. This issue is reproducible in Themelios and it can be used to test the proposed fix. Using this we can also begin to research the next step to weakening consistency further, focusing on write consistency.

Acknowledgement

This project has received funding from the EUROPEAN HEALTH AND DIGITAL EXECUTIVE AGENCY (HADEA) program under Grant Agreement No 101092950 (EDGELESS project).

Bibliography

[1] "Kubeedge." Accessed: Feb. 16, 2024. [Online]. Available: https://kubeedge.io/

[2] "K3s." Accessed: Feb. 16, 2024. [Online]. Available: https://k3s.io/