Container auto-scaling system
using sliding window regression with fuzzy entropy

2030416 Naoya Yokoyama

When offering services such as e-commerce on the cloud, there is a necessity to modify the amount of server resources provided in accordance with the irregularly increasing and decreasing traffic. This need arises when there's a desire to maintain a constant level of service while, at the same time, doing one's utmost to restrain costs.

There is an abundance of prior research in fields like time series forecasting and load prediction. Many of these approaches rely on traditional time series forecasting, which necessitates that the data used for learning adhere to stationary or unit root processes, or they use deep learning approaches that involve using a vast amount of data and parameters. Given the high demand for data and time in learning and inference, it proves difficult to provide the necessary server resources for burst traffic, which can drastically increase or decrease within a span of minutes.

On the other hand, there are preceding studies that use approaches such as sliding window learning, which involve partitioning data finely and repeatedly conducting learning and inference in a sequential manner. Existing auto-scaling methods that apply this sliding window learning take into consideration not only the most recent load but also burst traffic, thus providing server resources needed in the near future.

In this study, we propose a traffic forecasting method that involves dynamic window size changes that can follow even slight trend changes. This method is based on regression estimation using sliding window learning and incorporates burst traffic detection using fuzzy entropy. Furthermore, we propose a new auto-scaling system that applies decision regression trees to multiple load metrics.

As a platform for operating this system, we adopted container virtualization technology. Container virtualization is a technology that allocates server resources independent from other processes for a single process. It enables the flexible allocation of computer resources on the server to processes as needed. Compared to the conventional method of building one application server per virtual machine, using container virtualization can reduce the start-up time of an application server to a matter of seconds.

In the evaluation, we conduct four comparative experiments using actual traffic rather than simulations. In the comparative experiments, we use traffic data publicly available on the web to reproduce traffic patterns with a load generator and output to the system under experiment. As baseline methods,

we compare with two prior studies and the auto-scaling feature of Kubernetes, known as Horizontal Pod Autoscaling, which is the de facto standard as a container orchestration tool.

As a result, compared to the baseline methods, the proposed method reduced the number of request failures and improved the Mean Squared Error (MSE) between the ideal container count and the actual container count by an average of 150.17 points.