Using Redis one-way gates to eliminate massively parallel high speed race conditions

Author: Peter Gaivoronski

Claps: 149

Date: Sep 23, 2019

At Klaviyo we send billions of emails per month, so we often run into problems of scale in the strangest of places. In order to send out this many emails and then process the results, we have to massively parallelize our data flows. When sending emails, we break up each group of emails (for a campaign or flow) into many batches, and process those asynchronously using workers consuming from task queues. Because each worker is not aware of what the other workers are doing, this often leads to race conditions when the workers try to access and modify the same centralized data stores at the same time.

The Klaviyo platform has a system called â€œSmart Sendingâ€, which allows a sender to specify a number of hours during which a customer can only receive one promotional email. This means that no matter how many emails are scheduled to go to the customer, only one of those emails can actually be sent to the customer during that window. Our users may schedule many campaigns simultaneously that include overlapping customer segments, which means that a person can easily be queued up to receive 10 or more emails at the same time, and the system has to make sure that only one of those emails actually sends.

One way to try to solve this issue is to serialize the workersâ€™ access to the database. The task that transactionally locks the customer row first can do the send and any other tasks that attempt to access the customer have to get in line behind it. Once the other tasks do get access to the row, they see that the customer has already been sent to recently and therefore should be skipped. This approach works in theory and we commonly use it for small-scale jobs. Our customer databases are very large, however, and we send a very large number of emails per second at peak. This means that locking individual rows would generate a very large number of transactions on the database very quickly, which would slow down our database servers to a crawl. Sending speed is extremely important to us, so we have to look at other approaches.

Other ways in which we can do this include a centralized locking system from which we can request a certain customer per task, or a write-only system where we can have workers compete and only one will win the race, whileâ€¦