Introduction to Computer Science for Engineers

Load Balancing - Random Queue

This assignment **closed** December 8, 2024 at 23:15.

A *resource sharing* system involves a large number of loosely cooperating *servers* that want to share resources. Each server agrees to maintain its own **queue** of items for sharing, and a central authority distributes the items to the servers (and informs users where they may be found). For example, the items might be songs, photos or videos to be shared by a large number of users. To fix ides, we will think in terms of millions of items and thousands of servers.

We will consider the kind of program that the central authority might use to distribute the items, ignoring the dynamics of deleting items from the systems, adding and deleting servers, and so forth.

If we used a *round-robin* policy, cycling through the servers to make the assignments, we get a balanced allocation, but it is rarely possible for a distributor to have such a complete control over the situation: for example, there might be a large number of independent distributors, so none of them could have up-to-date information about the servers. Accordingly, such systems often use a *random* policy, where the assignments are based on random choice. An even better policy is to choose a random *sample* of servers and assign a new item to the server that has the fewest items. For small queues, differences among these policies are immaterial, but in a system with millions of items on thousands of servers, the differences can be quite significant, since each server has a fixed amount of resources to devote to this process. Indeed, similar systems are used in Internet hardware, where some queues might be implemented in special-purpose hardware, so queue length translates directly to extra equipment cost. But how big a sample should we take?

Assignments

1. To implement a load balancing simulation we first need a `RandomQueue`. This class derived from `Queue`. The RandomQueue supports the following methods:

method	return type	semantics
`is_empty()`	`bool`	Is the random queue empty?
`enqueue(item: str)`	`None`	Add `item` to the queue
`dequeue()`	`Optional[Any]`	Remove and return one random item from the queue (sampling without replacement)
`sample()`	`Optional[Any]`	Return one random item from the queue

Hints

- Have a look into the lectures `MyStack` implementation. You can change it to a Queue.

Template files

Get all files in an archive templates.zip or templates.tgz.

- `icse_queue.py`
- `random_queue.py`

Miit Dholakia | | | |