Lab 3. Redis

- what
- why
- proc/cons
- use cases
- features

Exercise

- install
- run
- connect
- simple queries
- pub/sub
- ???

Lab 3

Problem

Imagine, you have a source of tasks, e.g. an incoming stream of events and every event has to be handled (and this process is relatively long, like sending an email, downloading an image or a video, running a deep learning face recognition system and so on). And for some reason (legacy/speed of development etc.) you system is written in one of the most popular scripting languages (PHP/Python/Ruby/..) that don't support true parallelism of threads (due to GIL).

The problem is that even if you would love to parallelize these tasks (hopefully indepentent of each other), you cannot achieve true parallelization, so for some reason you decide to write your own (and not to use the already existing solutions) system that enables parallel execution of a python script.

This script's purpose is to get a task and execute it. Here where Redis shines - it allows us to create a part of the system that we will use as an independent atomic queue. Our clients will submit tasks to this queue and our workers will simply pull the tasks from the queue. In theory, this system can scale pretty well, but there are always caveats and corner stones, yet it is still viable for simple web crawler paralellization and so on.

We will create a basic client/worker scripts that will communicate via Redis.

Plan

1. Create a script that connects to Redis (see python redis package)
2. Using the connection from step 1, get your hands on the api the package provides and try some simple commands like `SET`, `GET`, `DEL`
3. Write a script that always waits (blocking call `lpop` or `rpop`) for an element to pop from the list (Redis provides an interface to use lists and queues via push and pop)
4. And finally, write a script that submits tasks (anything) to the queue so that workers can atomically access the queue and do the job

Outcome

We hope that you will be able to write a system that is small yet powerful to scale and later, if you need, you could run multiple servers with hundreds of workers that will all listen for tasks from Redis queue to perform some big data blockchain ai bitcoin hype analysis, or send spam emails, or at least something truly worthwhile

You will also remember what is a blocking call and IPC from the OS course

Miscellaneous

- Run Redis locally (requires Docker)

```
docker run --name local-redis -d -p 6379:6379 redis:alpine redis-server --
appendonly yes

# `local-redis` -- the name of the container created
# `-d` -- run the script in background, i.e. daemon mode
# `-p 6379:6379` -
- make port 6379 from the host (of the docker) be forwarded onto the port 6379
in the docker container
# `redis:alpine` -- a lightweight redis image
# `redis-server` -- a redis server executable
# `--appendonly yes` -
- a flag that enables Redis to persist data on the file system
```