Dr. Josep Sampé/Dr. Marc Sanchez-Artigas
`{josep.sampe, marc.sanchez}@urv.cat`

Distributed Systems 2019/20

Task#2 – Mutual Exclusion

Guidelines

- Please submit your solutions until **Wednesday, May 27, 23:55 PM** through Moodle. Only electronic submission is accepted. Please only use PostScript or PDF. We must be able to print out your submissions on a standard monochrome printer so that they are fully readable.
- This assignment can be solved in **teams of two students**.
- By submitting any processed exercise you hereby commit to Ethics rules promulgated by URV (see http://wwwa.urv.cat/la_urv/3_organs_govern/secretaria_general/legislacio/2_propia/comunitat/estudiants/inst_regl_reg_disc_est.pdf). This means that you should try to write the answers in your own words. However, if you use external materials, you have to cite them correctly.

1. Overview

The goal of this assignment is to implement a distributed version of a simple algorithm to ensure *mutual exclusion*, i.e., that **exactly one** cloud function can enter a critical section at any given instance of time. In the particular case of this assignment, *mutual exclusion* will be used to protected access to a shared file stored in IBM COS.

1.1 General architecture of the system

In this architecture, there will be one single master function and *N* slave functions, which would act on parallel requesting **write permissions** to the master function.

1.2 Implementation technologies

For the implementation, you MUST use **IBM-PyWren**[1] [1]. **IBM-PyWren** is an open source project that allows the execution of Python code at scale on top of IBM Cloud Functions, i.e., IBM's Functions-as-a-Service (FaaS) platform, based on Apache OpenWhisk. PyWren delivers the user's code into the serverless platform without requiring knowledge of how functions are invoked and run.

2. Mutual Exclusion Problem

2.1 Objective

The goal of this problem is to provide a distributed implementation of a mutual exclusion algorithm using IBM COS. There will be one master function and a variable number of slave functions. All the slave functions will update a common **shared** object called "`result.json`", appending into this file the `id` of the slave function when it gets permission to write.

2.2 Correctness

The master function must return a list of IDs in the order that he granted the "write" permissions to slaves. Slave functions will not return anything. At the end of the execution, your program will have to read the "`result.json`" file from COS and check whether it follows the same order as that of the list returned by the master function. This will mean that your implementation of mutual exclusion is **correct**.

2.3 Slaves

To ask for write permission, the slave function with identifier `id` will have to store a special file called "`p_write_{id}`" into COS. Next, it will have to wait until the master function writes back a special object called "`write_{id}`" into the same bucket, which will indicate the slave function that has been granted permission to **exclusively** write to the "`result.txt`" file. As IBM COS do not support object updates in place, a write will entail to first read the "`result.txt`" file from IBM COS, append the `id` of the slave function, and insert this file back to COS. Hence, mutual exclusion will be used to guarantee **atomic** writes to this file.

1. IBM-PyWren - PyWren on IBM Cloud, URL: https://github.com/pywren/pywren-ibm-cloud

2.4 Master

The master function will be periodically monitoring COS waiting for new “`p_write_{id}`” objects. If new write permission request objects “`p_write_{id}`”s are available, it will sort them by time of creation, pop the first one, and store the corresponding write permission grant object “`write_{id}`” back to IBM COS. Once the write permission is granted, the master will wait until the “`result.json`” file has been updated, which will mean that the slave function has already appended its `id` at the end of the file. Only then, it will be **safe** for the master function to select the oldest “`p_write_{id}`” object remaining in the bucket and grant a new write permission. This loop will continue until there will be no more “`p_write_{id}`” objects in COS.

NOTE that any given time, it could only exist one single “`write_{id}`” file in IBM COS, which signals that exclusive access has been granted to just a single slave function at a time. This guarantees the correctness of the algorithm.

2.5 Code Skeleton

Here you can find a skeleton of the code to help you solving the assignment.

```
N_SLAVES = 5

def master(x, ibm_cos):
    write_permission_list = []

    # 1. monitor COS bucket each X seconds
    # 2. List all "p_write_{id}" files
    # 3. Order objects by time of creation
    # 4. Pop first object of the list "p_write_{id}"
    # 5. Write empty "write_{id}" object into COS
    # 6. Delete from COS "p_write_{id}", save {id} in write_permission_list
    # 7. Monitor "result.json" object each X seconds until it is updated
    # 8. Delete from COS “write_{id}”
    # 8. Back to step 1 until no "p_write_{id}" objects in the bucket

    return write_permission_list


def slave(id, x, ibm_cos):
    # 1. Write empty "p_write_{id}" object into COS
    # 2. Monitor COS bucket each X seconds until it finds a file called "write_{id}"
    # 3. If write_{id} is in COS: get result.json, append {id}, and put back to COS result.json
    # 4. Finish

    # No need to return anything


if __name__ == '__main__':

    pw = pywren.ibm_cf_executor()
    pw.map(slave, range(N_SLAVES))
    pw.call_async(master, 0)
    write_permission_list = pw.get_result()

    # Get result.json
    # check if content of result.json == write_permission_list
```

1. IBM-PyWren - PyWren on IBM Cloud, URL: https://github.com/pywren/pywren-ibm-cloud

2.6 Tasks

Here you can find the list of tasks to address:

1. **[7 points]** Implementation of the mutual exclusion algorithm using COS. The code should allow changing the number of *N* slave functions.
2. **[0.5 points]** Experimentation with different values of *N.*
3. **[1 point]** Satisfactory documenting of the solution and results.
4. **[1.5 points]** Answer the following questions on mutual exclusion:
    a. Many distributed algorithms such as the one proposed in this assignment require the use of a **master** or **coordinator** process. To what extent can such algorithms actually be considered distributed? Discuss
    b. Now suppose that the **master function** crashes. Does this always violate the correctness of the algorithm? If not, under what circumstances does this happen? Is there any way to avoid the problem and make the system able to tolerate coordinator crashes? Discuss
    c. In the proposed algorithm, write permissions are granted in the order in which are requested, so no slave function waits forever **(no starvation**). If the master function chose the slaves functions **randomly**, then could slave functions suffer from starvation?

3. Submitting

Please, submit a three-page document describing your proposed solution along with the plots and the answers to the theoretical questions. Further, an in-depth explanation of the results is MANDATORY. The document should also include full references for the papers and other sources that you have consulted. **If you are working in a group, you must say who has been responsible for each portion of the work.**

Personal github. All groups must create their own `github` repository with their code and small documentation.

Finally, create a `.zip` file called `DS_A1_name1_name2.zip`, where `name1` and `name2` will be the family name of each component of the group. Include in the `.zip` file the document in `.pdf` format along with all the Python code.

References

[1] - Josep Sampé, Gil Vernik, Marc Sánchez-Artigas, Pedro García-López, "Serverless Data Analytics in the IBM Cloud". Middleware Industry 2018: 1-8

1. IBM-PyWren - PyWren on IBM Cloud, URL: https://github.com/pywren/pywren-ibm-cloud