17CS352: Cloud Computing

Class Project: Rideshare

Date of Evaluation: 16-05-2020
Evaluator(s): Prof. Venkatesh Prasad
Submission ID: 1248
Automated submission score: 9

SNo	Name	USN	Class/Section
1	Shivangi Raj	PES120170098	C
2	Prathusha Kunka	PES1201701831	C
3	Meghana Arumilli	PES1201701279	C
4	Ajay Chavan	PES1201701649	D

Introduction

- Database-as-a-Service (DBaaS) is a PaaS component that dramatically simplifies the provisioning, configuration and management of databases in an enterprise. DBaaS allows IT organizations to provide databases with an easy-to-consume, self-service user experience.
- It ensures that these databases are operated in a manner that maintains data security and privacy. End users benefit from improved agility, application reliability and performance.
- Orchestrator is used to manage interconnections and interactions among workloads on private and public cloud infrastructure.
- RabbitMQ: Supports multiple messaging protocols which acts as a broker that send data to queues with customized persistent level
- Zookepper: A centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. All of these kinds of services are used in some form or another by distributed applications. It also checks for all the events handled.
- We are using 6 containers:

  Ride, User, Orchestrator, Master, Slave and Shared Data.

Related work

Online videos:

Pdf's links:

- http://zookeeper.apache.org/
- https://www.rabbitmq.com/getstarted.html
- https://docker-py.readthedocs.io/en/stable/

Stack overflow has helped us for rectifying our errors.

ALGORITHM/DESIGN

Algorithms implemented:

- Zookeeper
- Queues:
  1. Sync Queue: As master db gets updated so as that slave's db should also get updated. For the event to occur we use persistent db so that the data present in master db gets copied to slave's db.

2. Response Queue: Queue's up all the response sent the slaves on read request from the orchestrator.
3. Read Queue: A queue that reads the data of the slave on read request from the orchestrator.
4. Write Queue: A queue that writes the data into the master with a write request from the orchestrator.

- Scaling: it's a process of auto scaling the slaves. This can be achieved by calling time function every 2 minutes and calculating total count http request/total slaves. Based on this calculation we either scale up the containers or scale down. If number of containers are more than slaves the container are then deleted

Setup of zookepper:

Zookeeper is to maintain a watch over each container. It keeps a check on each event occurred and also handles it. We achieved this by using the client. ChildrenWatch function of the kazoo library. Whenever a child node of the path "orchestrator" was created or deleted, the ChildrenWatch function was called and at that point, we checked the number of children nodes currently running against the number of nodes that are supposed to be running. If there is a deficit, we then create a new child node. The child node is then assigned a PID and a slave. The PID is then added to the list of child nodes.

TESTING

1. Initial times of our project submission we got load balancer is not working so we couldn't figure out what was the problem but then got to know that what api's they were sending were wrong. So we then checked all api's in postman and then the load balancer got worked.
2. After getting 9 we couldn't figure out what was the error.

CHALLENGES

1. Understanding concept especially zookepper, tutorials and figuring them out.
2. Creating link of databases between master and slave using Sync queue
3. All slave's data didn't get updated, so we created a folder which contained persistent database i.e. whenever a data is stored in master then that data is sent to slave using sync queue and then copy persistent db is copied and pasted to slave .

Contributions

Each individual team member must list their contributions towards it.

Names	Contribution
Shivangi Raj	Implemented crash api's, spawning, Zookeeper's container image, error handling and setting connections between orchestrator, ride and user
Prathusha Kunka	Checking of orchestrator's requests send to read the slave and send backing its response of all the api's to orchestrator, and all the changes done in ride and user instance.
Ajay Chavan	Implemented sync queue between master db and persistent db and storing persistent db's content and coping the data to slave db .
Meghana Arumilli	Creating Write queue in orchestrator which does all the write operations and store data in master db

CHECKLIST

SNo	Item	Status
1.	Source code documented	Commented and cleaning of code is been done
2.	Source code uploaded to private GitHub repository	https://github.com/Shivangi-Raj/spam_cloud_computing.git
3.	Instructions for building and running the code. Your code must be usable out of the box.	1.For running orchestrator part: • cd cc_project • sudo docker-compose build • sudo docker-compose up 2. For running ride instance: • cd ride • sudo docker-compose build • sudo docker-compose up 3. For running user instance: • cd user • sudo docker-compose build • sudo docker-compose up 4. For running api's in postman: • http:// user-rides-659012964.us-east-1.elb.amazonaws.com / {api's given} 5.For crash api's/worker list/clear db:

		• http://34.194.180.47:80/{api’s given}