CS695 Virtualization and Cloud Computing
Assignment 4

Vedang Asgaonkar 200050154
Sanchit Jindal 200020120

Contents

1 Problem Description

A serverless architecture include providing a DAG (Directed Acycylic Graph) of functions that trigger when certain conditions are met. We can connect the functions together creating a workflow which will run automatically.

We try to recreate the functionality provided by applications like nextFlow, in which we can specify the inputs and outputs for each function, and as all inputs to a function are available it runs and produces the output

2 Approach

The Repo is available at `https://github.com/sanchit1053/nextNextFlow`

2.1 Idea

To create a workflow the user needs to create a `xml` file that defines the functions and the channels that connects the functions

The program runs the script and passes the data from the channels to the correct processes and when the processes have all the needed data, they run and output data to the specified output channels

2.2 Components

2.2.1 Channels

- In our implementation of the Serverless architecture we connect the different processes using files which we call channels
- The channels act as queues from which the process can take inputs sequentially and release outputs to
- The channels provide a generic medium to transfer data between the functions and also act as the triggers in the program
- We can initialize the channel with some default data or leave it blank

2.2.2 Processes

- Each process is defined as the input channels, output channels, and the commands that will be run
- In our implementation the user needs to manually read the file that it takes as input to get the data and then put the data into the output data
- The script can be given directly in the xml file or we can point to another file

2.2.3 Mux

- Mux is a connector that directly connects the channels to one another
- It can be used to duplicate data in a channel along multiple channels or to combine from multiple channels to one

2.3 Main Loop

- For each process we check each of the input channels and if all of the input channels have data then we start the process execution
- To setup the execution we create a docker container with the process name
- We create the input and output channels (represented as files with same name) to the docker container
- One unit of data that is stored in the input channels is added the corresponding files in the container
- The process is started inside the container in a child process

- We setup a signal handler to check when the child returns, and when it returns the processes output is collected from the container to the channels
- We check for each Mux and move the data along the channels

3 Further Work

This is a start to a full-fledged software architecture and many more features can be added to the application

- The ability to write the workflow in other languages such as python, or javascript instead of only bash
- Providing even better parallelization to the application to run move data between channels faster and less time consuming
- being able to move variables through the channels directly instead of the user to manually work with files to get the data
- backend features such as live-migration and resource reallocation to the different containers to provide better availability