Philz Coffee Process Simulation

Team:

RedHat

CMPE 275 Project2/Spring 2015

By

Anup Dudani (009991007)

Meghana Gogi (009801051)

Prabhu Maniraj (010014394)

Sai Sudha Kiran Reddy Dwaram (009999197)

Contents

1. Introduction

Philz coffee real time scenarios have been considered to model our project simulation using Simpy. Here we are simulating the Philz coffee real time scenarios like customers in the queue, the scenario in which people who are serving the coffee for customers, the scenario in which the customers are waiting for coffee deliver, and the scenario in which the customer interact with the store people for coffee bill in order to checkout, different type of coffee's etc. These actors and the behaviours will be explained in detail later. These scenarios you can see in the real time as images below:

Three serving counters is considers as an actor, the counter interaction in which the coffee is being served to the customers.

In the below image, the check out counter is been considering. Where the customer interaction will happen in order to pay the bill for orders

In the below picture, It is observed that the coffee bean will be refillied once it is done. This scenario is been considered and the time taken for refill the beans also been considered in order to do simulation model

The coffee refilling time is calcuated based the orders for Dark roast, Light roast and Medium rost. We have considred coffee beans usage as more for light roast and the medium usage of beans for medium roast and the least usage of beans for dark roast.

In the below picture, the customers waiting time for the order delivery has been considered during modelling.

People waiting in the queue and the patience level of people to be in the queue (like people might leave the queue in the middle) and the people interaction with the service counter people to order the coffee has been considered in our project, which can be observed in the below picture.

2. Technologies Used

Python

It is object-oriented, high-level programming language. It has high level built in data structure features. Python is simple, easy to learn the python syntax, maintenance cost is less for programming, code can be reused. It is an interpreter standard library. Python has many advantage which is suitable for project simulation. This is the reason we opt this programming language.

Simpy Package

Simpy is discrete-event simulation framework based on python. It is it used for asynchronous networking communication and to implement multi-agent systems. There are various python generator function are available in simpy which can be used to model the scenarios of customers, interaction etc. It even provides various shared resource for checkout counters etc. The same implementation is coming in our project as well. Hence we choose simpy for implementation.

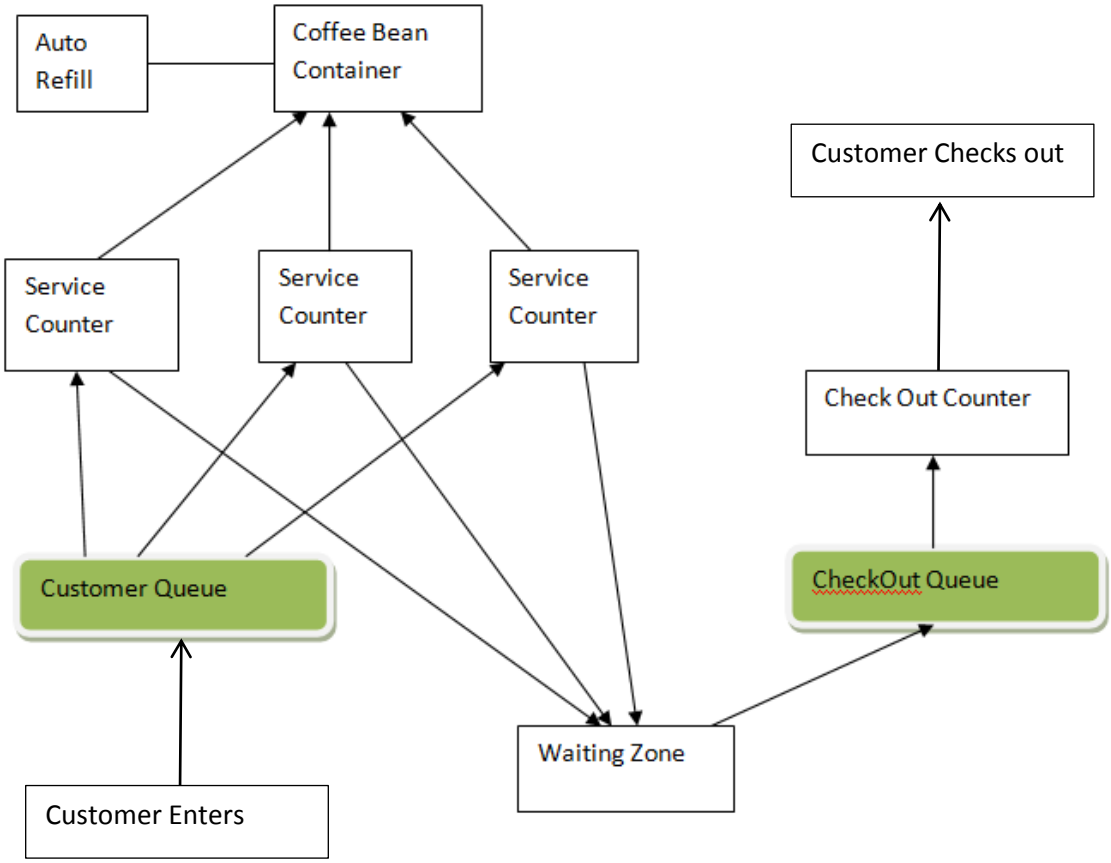
Elasticsearch, Logstash and Kibana (ELK Stack)

Elastic search is basically a search server, which is used for the deep searching and data analytics.

Logstash is for centralized logging, log enrichment and parsing. Grok filtration is done to get a cleaned data from the csv file using logstash

Kibana is for data visualization.

Architecture



This system is built based on process-based discrete-event simulation model using SimPy:

Simulation time is considered to develop the system which can be given dynamically by the user. In this system, based on our observation user can come to the shop at any random time and in any number, this makes us produce customers randomly. Number of counters which serves the customer can be modified in the simulation model to try different scenarios.

There are 3 different processes running asynchronously

1. To which servers the customers in real time the person who prepares coffee.
2. A process which checks the bean container level and refill if the bean level goes below the threshold value.
3. A process which handles the checkout process.

Considering Queues the system has 2 queues:

1. The queue where user first enters the shop and joins which leads to service counter. And there is a sub process embedded in the first queue, if the user lost his patience level he can move out the shop by breaking the queue.
2. When the user has got his order prepared by the coffee vendor, then he/she moves to the checkout counter, basically it is first come first serve mechanism which is a queue obviously.

Waiting Time:

Waiting zone is the place where customers will wait until the coffee is being prepared by the vendor.

3. Actors and behaviours in details

3.1 Actors

1) Customer
2) Service counters
3) Checkout counter
4) Bean Container

Customer: Customer comes to Philz Coffee to get his desired coffee. He has few attributes and methods as follows, which are useful in the scenarios, which will be discussed in the following section:

Patience: Patience attribute is used to define how to long the person can wait for his number in the queue. If the person's patience is more, he can wait for more time.

Service Counter: Service counter is used to take order of Customer and provide the coffee.

Bean container: It is the place from where the refilling of service counter is done. Bean container has some capacity of coffee beans, if the capacity of Bean Counter is less than 10%, it will go for refill.

Checkout Counter: Where the customers will pay the coffee bill and this is the last interaction which happens before leaving the coffee shop has been considered in our project.

3.2 Interactions

Below interactions are considered in this simulation project:

Revoking the queue: When the incoming customer lost patience to stay in queue then he/she will come out of the queue and may move out of the shop.

Ordering: This is an interaction between the customer and the coffee service counter. This interaction will take some to take decisions on selecting coffee type (dark roast, medium roast and light roast) and to choose coffee size (small, medium or large).

Refilling of coffee bean container: This interaction is based on the level of coffee bean central container. There is container size is handled by a variable which keeps the track of container level when it goes to a threshold level then the refilling of the container starts. Level of the container will be reduced based on type of the coffee and size of coffee.

Checkout: Final interaction is between the customer and the checkout counter, where the user waits in the queue to finish the process and it's the final step.

4. Modelling using Simpy

For modelling the above Philz System we are using simpy package of python. Especially simpy.Environment(), simpy.Resource() and simpy.Container() which are explained below.

simpy.Environment()

```
env=simpy.Environment()

counter=simpy.Resource(env, capacity=2)
checkout_counter=simpy.Resource(env, capacity=1)
bean_container=simpy.Container(env, CONTAINER_SIZE,init=CONTAINER_SIZE)
env.process(container_control(env,bean_container))
env.process(source(env,INTV_CUSTOMERS,counter,bean_container,checkout_counter))
sim_time=input("Enter the Simulation time:")
env.run(until=int(sim_time))
```

Simpy.Environment() is used to create an environment for the entire simulation to take place. All the processes live in this environment. The processes interact with each other and with environment through events.

Simpy.Resource()

```
env=simpy.Environment()

counter=simpy.Resource(env, capacity=2)
checkout_counter=simpy.Resource(env, capacity=1)
bean_container=simpy.Container(env, CONTAINER_SIZE,init=CONTAINER_SIZE)
env.process(container_control(env,bean_container))
env.process(source(env,INTV_CUSTOMERS,counter,bean_container,checkout_counter))
sim_time=input("Enter the Simulation time:")
env.run(until=int(sim_time))
```

Simpy.Resource() is used where a resource is shared among multiple processes. In our case, resources are the Service Counters where customer orders coffee and Checkout Counter where customers checksout. It takes two arguments. One is the environment which consists of

processes and capacity which represents the number of resources(in this case service counters and checkout counter).

simpy.Container()

```
env=simpy.Environment()

counter=simpy.Resource(env, capacity=2)
checkout_counter=simpy.Resource(env, capacity=1)
bean_container=simpy.Container(env, CONTAINER_SIZE,init=CONTAINER_SIZE)
env.process(container_control(env,bean_container))
env.process(source(env,INTV_CUSTOMERS,counter,bean_container,checkout_counter))
sim_time=input("Enter the Simulation time:")
env.run(until=int(sim_time))
```

Simpy.Container() is used to create a container with some initial capacity. The arguments include init which takes the value of capacity of bean_container. The current level of bean_container can be achieved through bean_container.level().

Sample Output Screenshot:

```
Enter the Simulation time:100
Customer 0 arrived at  0.0000
Customer 0 patience is  2.228
 0.0000: Customer 0 Waited for  0.000
Customer 0 takes 13.221 time near counter
Customer 1 arrived at  3.8895
Customer 1 patience is  2.590
 3.8895: Customer 1 Waited for  0.000
Customer 1 takes 26.328 time near counter
13.2210: Customer 0 has finished ordering 3 small darkRoast
Bean_Usage is 15
Container level is 85
13.2210: Customer 0 wait_time after ordering is 9.000000
Customer 2 arrived at 16.3298
Customer 2 patience is  2.416
16.3298: Customer 2 Waited for  0.000
Customer 2 takes 10.966 time near counter
Customer 3 arrived at 17.5625
Customer 3 patience is  2.058
19.6201: Customer 3 left out  due to no patience after  2.058
Customer 0 paying bill at 22.2210
27.2961: Customer 2 has finished ordering 2 medium lightRoast
Bean_Usage is 40
Container level is 45
27.2961: Customer 2 wait_time after ordering is 10.000000
Customer 4 arrived at 29.0359
Customer 4 patience is  1.999
29.0359: Customer 4 Waited for  0.000
Customer 4 takes 18.447 time near counter
30.2176: Customer 1 has finished ordering 3 medium mediumRoast
Bean_Usage is 30
Container level is 15
30.2176: Customer 1 wait_time after ordering is 15.000000
Customer 2 paying bill at 37.2961
Customer 5 arrived at 43.5906
Customer 5 patience is  2.459
43.5906: Customer 5 Waited for  0.000
Customer 5 takes  6.353 time near counter
Customer 6 arrived at 45.0070
Customer 6 patience is  1.120
46.1270: Customer 6 left out  due to no patience after  1.120
```

Explanation of output:

First input is given regarding simulation time.

First Selection made above includes customer0 and customer1 arriving at service counters. Customer0 arrived at time 0.000 and he/she can use any of the available service counter(resource). So it is displayed that he waited for 0.000. Now he takes 26.328 time to order. In the mean while customer1 arrived and as there is another service counter available, he/she need not wait and they can access the service counter.Customer 0 has finished ordering at 13.2 and the Bean_Usage and Current bean_container level is displayed taking into account his/her order. The bean_container is refilled when container.level <10% of original. The other selection above shows Customer3 left out as he has less patience when customer1 and customer2 are ordering.

5. Visualization Screen Shots

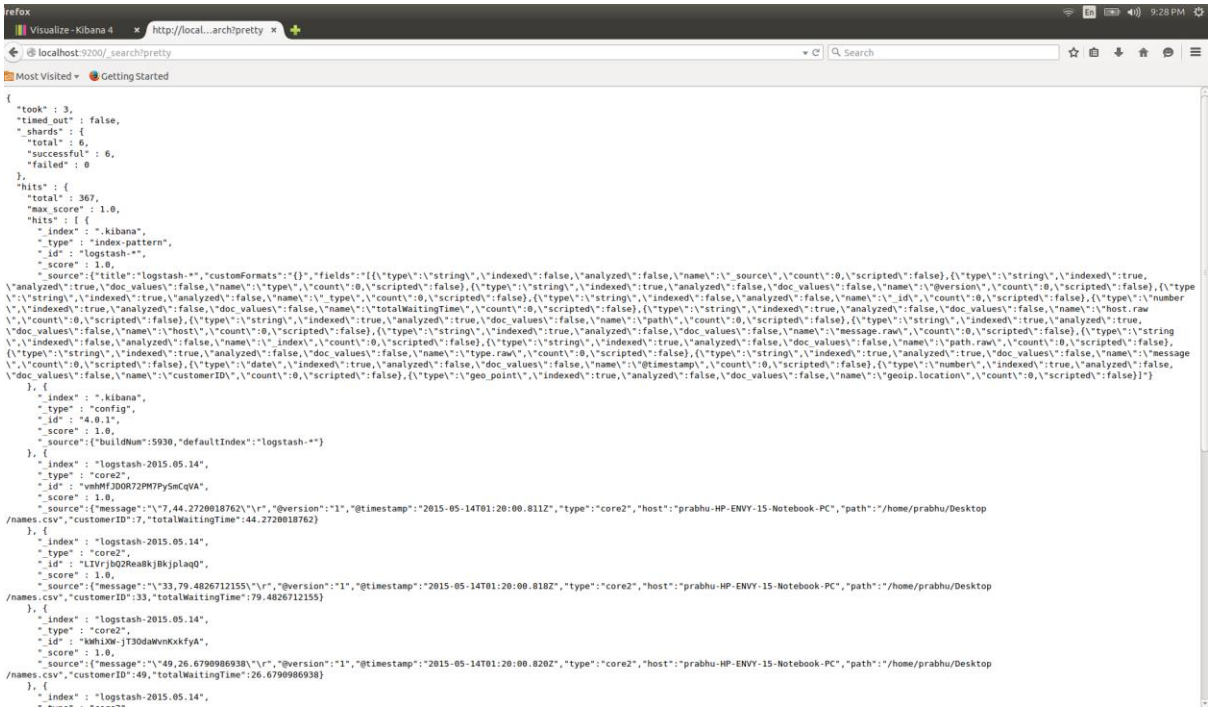
From the python code we are generating logs for total waiting time per customer in the coffee shop. The generated csv file is then parsed to JSON format using logstash which is pre-configured. There is configuration file called logstash-cmpe275.conf where there is pattern which will fetch the values from the csv file and dump it to the elastic search which is running on the localhost:9200. In elasticsearch.yml file the path for kiana is mentioned. Kibana listening to elasticsearch will produce visualization output based on users perspective.

The config file has a module for grok, which will select the desired pattern from the csv file and parse it according to users need. Here in our scenario we are taking the first and second field from csv as number and giving name to it as customerID and totalWaitinTime Respectively.

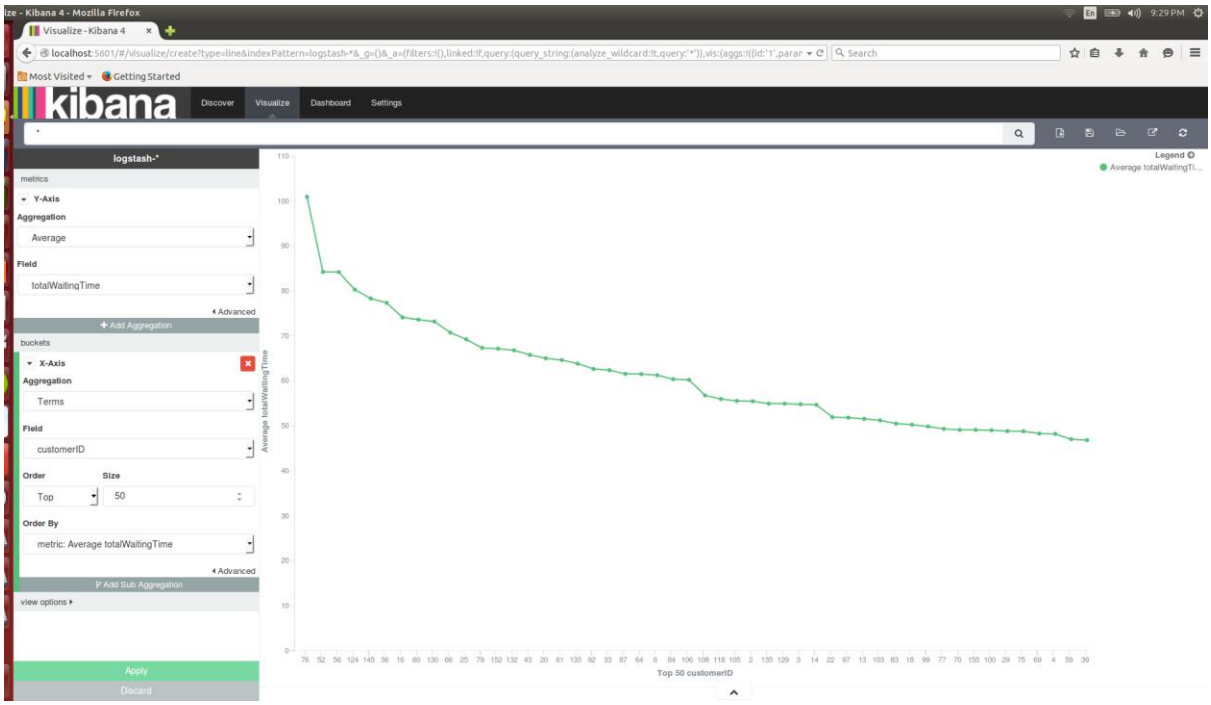
```
input{
new 2
file {
    path => "/home/prabhu/Desktop/names.csv"
    type => "core2"
    start_position => "beginning"
    sincedb_path => "_/null"
    }
}
filter{
    grok
    {
        match => ["message","%{NUMBER:customerID:int},%{NUMBER:totalWaitingTime:float}"]
    }

}
output {
    elasticsearch {
    port => "9200"
    embedded => true
    }
    stdout { codec => rubydebug }
}
```

https://localhost:9200/_search?pretty

Running this RESTcall will fetch information about the logs and display the recent 10 logs which is present in the elastic search. Here it's obvious that the message column of the each log has customerId and totalWaiting Time, which is mentioned in the grok pattern.



Kibana visualization for the top 50 customers whose waiting time are high and the graph shows the obvious details. Y- Axis has average waiting time and X-Axis has top 50 customer Ids. Kibana is listening on https://localhost:5601



6. Advantages and disadvantages of simulation models

Below are the few advantages for simulation models

- It can be used for quick investigation on effects of changes in real life situation that take place over many years.
- Complex system study can be achieved for investigation.
- Without the physical prototype we can build the product design.
- The real life situations which are dangerous can be investigate
- Cost is less, since we are just simulating the model rather building the complete project

Below are the few disadvantages for simulation models:

- It is difficult to produce mathematical model with poor knowledge on how physical system works. For instance like predicting tsunami or earthquake
- There are chances in inaccurate output in case the functions and formula used is in correct.
- Complex simulation might require large amount of memory usage and might requires fast processors.

7. Conclusion

Many things we have learnt from this project like

- Observing the real time scenarios and building the models based on observation.
- Scripting language python and the SimPy framework has been understood and learnt
- Learnt ELK stack and its working
- Visualising and analysing the project complexity

8. Referneces

[1] http://web.cs.mun.ca/~donald/msc/node6.html

[2] http://www.sv.vt.edu/classes/ESM4714/Student_Proj/class03/stillwater/prj/project/sim_pandc.htm

[3] http://simpy.readthedocs.org/en/latest/api_reference/simpy.core.html

[4] http://simpy.readthedocs.org/en/latest/topical_guides/events.html