

CS237 Group 7 Project Report:

Coupon Distribution System

Jiaqi Xiao #57047762

Yinhao He #14721976

Ananth Gottumukkala #37164068

Introduction

Online shopping has become one of the most frequent activities in our daily life. Merchants would provide some coupons for consumers to promote sales like Black Friday. Using coupons to stimulate consumption has become a common method for current businesses. However, it is not easy for normal consumers to manage all these coupons. In this project, we would like to develop a Coupon Distribution Platform for both consumers and merchants. Customers can easily get and manage their coupons on this platform. Merchants can directly distribute coupons.

The coupon distribution platform is based on the pub/sub messaging system which enables the users to subscribe to a certain type of coupons they would like to receive. In order to determine which pub/sub messaging system should be used, we did some research about the pub/sub messaging systems. We present the architecture of our system in this report. Also, we talked about how we used Kafka in our system. Multiple test results are shown at the end of this report.

We made an assumption that many customers rush to get a limited number of coupons at a certain time. In order to achieve that, we implemented the following features.

- Use Kafka to solve the problem of high-throughput data processing of a very large scale of messages.
- Use HBase to implement the storage of customers and coupons' information to ensure the read and write efficiency.
- After the implementation of our platform, use PostMan to test APIs.

Related Work

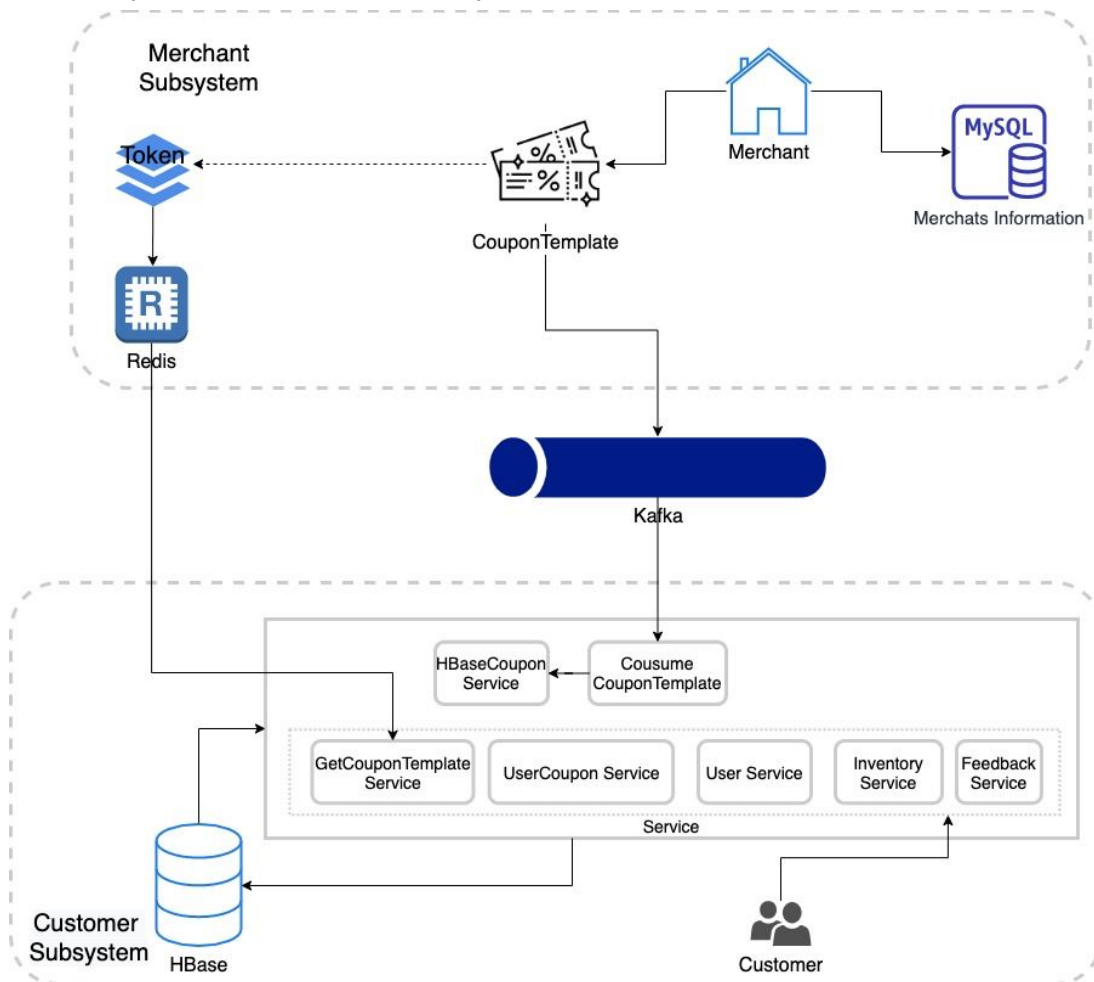
We have done research on multiple messaging platforms such as Kafka, RabbitMQ, ActiveMQ, Redis, etc. RabbitMQ and Kafka have a persistent and good throughput, while ActiveMQ is well suited for enterprise-level applications (Dobbelaere, 2017), while Redis is really fast. A study of various technologies, in particular RabbitMQ and Kafka, addressed the problem of increasing technology to use for our prototype. (Magnoni, 2015)

Back to our idea, Kafka might not be the most effective or the most practical one for us, but it's the most suitable one for now. After contrasting it with so many other message queue

structures, we have come to the realization that there are many explanations why we eventually chose Kafka. First of all, Kafka is a message queue system. In some time-limited activities, the application will generally be suspended due to a sudden increase in traffic and the application can not be processed. Second, Kafka can handle multiple producers seamlessly, whether they use multiple themes or the same subject. This makes the system ideal for platforms such as our coupon distribution platforms. It will be useful to retain Kafka's long-lasting message in our project. Messages are committed to the disk and will be stored under configurable retention rules. Flexible scalability and a wide variety of configuration choices in Kafka may be used to conduct several performance comparison tests to assess the specific results of various partitions and topic configurations. (Kreps, 2011) Finally, Kafka is the best option for our idea. Fast persistence, high throughput, replication mechanisms, and so on could provide massive support to the coupon distribution platform. Also, Kafka's high scalability enables us to perform some extra performance tests after the platform has been developed.

Design

In this section, the structure of the Coupon Distribution System is introduced, including Merchant Subsystem and Customer Subsystem.



Merchant Subsystem

The main functions of the Merchant Subsystem are the management of Merchant Information and the distribution of CouponTemplate. The basic information of Merchants is stored in MySQL. The Merchant can distribute any number of CouponTemplates to each Kafka topic. Some CouponTemplates require Token Verification, so the Merchant needs to upload Token Files in advance. These Token Files would be stored in Redis as a Set that is an unordered collection of Strings. Redis can provide high-speed read and write when a large number of coupons are consumed in a short period of time.

Customer Subsystem

The Customer Subsystem is responsible for all the services for customers. Customers can operate coupons and check their status. Also, there's a function for customers to feedback their comments on both coupons and the Distribution System.

When a customer is created, besides the basic information, UserId would be generated by joining the current total number of customers and a 5-digit random number. Customers can submit feedback on coupons and the system through Feedback Service.

With the annotation of '@KafkaListener', ConsumeCouponTemplate Service will keep listening to the Kafka topics and consume the coming messages of CouponTemplate. After receiving the CouponTemplate, it will be transmitted to a value object in Java for further processing. HBaseCoupon Service will persist all these CouponTemplate in HBase. Customers can check all the available CouponTemplate by Inventory Service. According to the title of CouponTemplate in the response of Inventory Service, customers can request that coupon with GetCouponTemplate Service. For a normal CouponTemplate, a row of data that contains the CouponTemplate and customer information will be inserted into the 'cp: coupon' table in HBase after verification of validate time and number limitation. However, if the CouponTemplate requires a token, then the token stored in Redis must be popped and verified. Each time one this kind of CouponTemplate is obtained, one token will be used and eliminated. When the tokens for a CouponTemplate are empty then the CouponTemplate is no longer available.

As soon as the user gets some valid coupons, the usable coupon information is available through UserCoupon Service. When a customer uses a coupon, UseCoupon Service will find the coupon by UserID and CouponTemplateID then updates the information in 'cp:coupon' to set it as used. Customers' all coupons could be viewed by two status, used or unused.

Kafka

We used Kafka in this project as the service which allows merchants and customers to exchange coupon data. A Kafka template is used to publish messages (coupon data populated by the merchant) to a specific topic corresponding to for example the type of coupon. A Kafka listener is configured to consume messages from the corresponding Kafka topic. Our implementation used a Kafka listener to consume coupon data and relay it to HBase for data persistence.

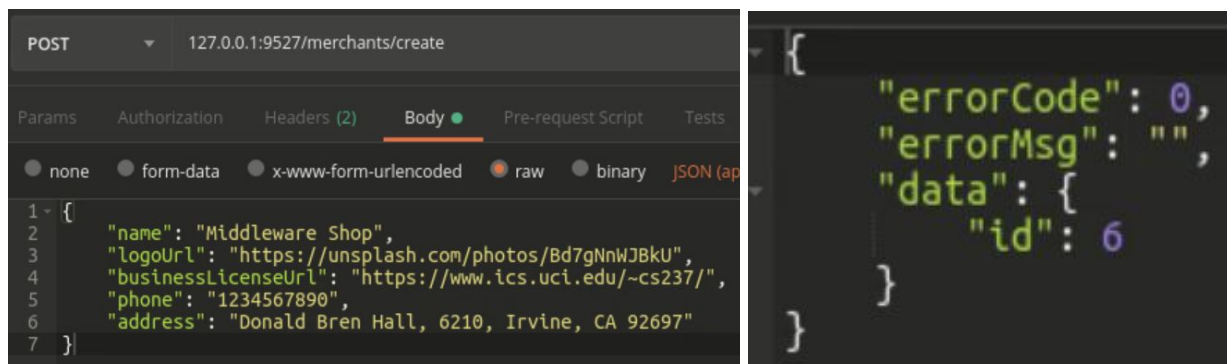
In addition to the HBase listener (GetCouponTemplateService), we also implemented additional listeners for each customer so that they can consume coupons in different topics asynchronously directly from Kafka. Unlike the HBase listener, which keeps listening and consuming data from Kafka topics, the customer listeners consume data from the appropriate topic only when the User specifically requests/uses a coupon. The coupon transaction history as well as the coupon validation and verification information is also stored locally by each User. In terms of implementation, the customer listeners are still a work in process (as of writing this report) and are being debugged so for now we have the customers consuming data from HBase since the implementation was simpler to get working end to end.

Results

We design a whole experiment to test the functionality of our Coupon Distribution System. We use Postman to test all the APIs in our system with HTTP requests. Click here to watch the testing video.

Create New Merchant

When the new merchant is created, it will generate an id that is auto-incremented in MySQL.



The image shows two side-by-side screenshots from a testing tool. The left screenshot displays a POST request to the endpoint `127.0.0.1:9527/merchants/create`. The request body is a JSON object with the following fields: `"name": "Middleware Shop", "logoUrl": "https://unsplash.com/photos/Bd7gNnWJBkU", "businessLicenseUrl": "https://www.ics.uci.edu/~cs237/", "phone": "1234567890", "address": "Donald Bren Hall, 6210, Irvine, CA 92697"`. The right screenshot shows the response body, which is a JSON object: `{"errorCode": 0, "errorMsg": "", "data": {"id": 6}}`.

Get Merchant Information with Merchant ID

```
GET 127.0.0.1:9527/merchants/6

{
  "errorCode": 0,
  "errorMsg": "",
  "data": {
    "id": 6,
    "name": "Middleware Shop",
    "logoUrl": "https://unsplash.com/photos/Bd7gNnWJBkU",
    "businessLicenseUrl": "https://www.ics.ucl.edu/~cs237/",
    "phone": "1234567890",
    "address": "Donald Bren Hall, 6210, Irvine, CA 92697",
    "isAudit": false
  }
}
```

Merchant Distribute CouponTemplate

```
POST 127.0.0.1:9527/merchants/distribute

Params Authorization Headers (2) Body Pre-reqs
none form-data x-www-form-urlencoded raw

1 - {
2   "background": 1,
3   "desc": "Description: Middleware",
4   "end": "2020-06-30",
5   "hasToken": false,
6   "id": 6,
7   "limit": 1000,
8   "start": "2020-03-01",
9   "summary": "Summary: Middleware course",
10  "title": "Middleware discount-1"
11 }

hbase(main):012:0> scan 'cp:coupons'
ROW COLUMN+CELL
fb0a1150d23cae0266d0 column=b:background, timestamp=1591769671869, value=\x00\x00\x00\x00\x00\x00
264ad55af119
fb0a1150d23cae0266d0 column=b:desc, timestamp=1591769671869, value=Description:
264ad55af119 Middleware
fb0a1150d23cae0266d0 column=b:has_token, timestamp=1591769671869, value=\x00
264ad55af119
fb0a1150d23cae0266d0 column=b:id, timestamp=1591769671869, value=\x00\x00\x00\x00
264ad55af119 06
fb0a1150d23cae0266d0 column=b:summary, timestamp=1591769671869, value=Summary:
264ad55af119 Middleware course
fb0a1150d23cae0266d0 column=b:title, timestamp=1591769671869, value=Middleware
264ad55af119 discount-1
fb0a1150d23cae0266d0 column=c:end, timestamp=1591769671869, value=2020-06-29
264ad55af119
fb0a1150d23cae0266d0 column=c:limit, timestamp=1591769671869, value=\x00\x00\x00
264ad55af119 0\x00\x00\x00\x03\xE8
fb0a1150d23cae0266d0 column=c:start, timestamp=1591769671869, value=2020-02-29
264ad55af119
1 row(s) in 0.1450 seconds
```

Upload Token File

The Token File contains several lines of tokens. Every time when a customer needs to get a coupon that requires a token in the inventory, one line token in Token File will be consumed. Check if the Token File is in Redis

```
127.0.0.1:6379> keys *
1) "bfcba139280545fe043c69ad2eda706"
127.0.0.1:6379> smembers 'bfcba139280545fe043c69ad2eda706'
1) "token-3"
2) "token-2"
3) "token-1"
```

Create New Customer

```
POST 127.0.0.1:9528/coupon/createuser

Params Authorization Headers (2) Body ● Pre-request Script
● none ● form-data ● x-www-form-urlencoded ● raw ● binary

1 - {
2 -   "baseInfo": {
3 -     "name": "Jiaqi Xiao",
4 -     "age": 23,
5 -     "gender": "m"
6 -   },
7 -   "otherInfo": {
8 -     "phone": "1234567890",
9 -     "address": "Inner Ring Rd, Irvine, CA 92697"
10 -  }
11 - }
```

```
hbase(main):015:0> scan 'cp:user'
ROW COLUMN+CELL
\x00\x00\x00\x00\x00\x02\x1FL column=b:age, timestamp=1591776221854, value=\x00\x00\x00\x17
\x00\x00\x00\x00\x00\x02\x1FL column=b:gender, timestamp=1591776221854, value=m
\x00\x00\x00\x00\x00\x02\x1FL column=b:name, timestamp=1591776221854, value=Jiaqi Xiao
\x00\x00\x00\x00\x00\x02\x1FL column=o:address, timestamp=1591776221854, value=Inner Ring Rd, Irvine, CA 92697
\x00\x00\x00\x00\x00\x02\x1FL column=o:phone, timestamp=1591776221854, value=1234567890
1 row(s) in 0.0090 seconds
```

Get Inventory Information (Available CouponTemplate for Customers)

```
GET 127.0.0.1:9528/coupon/inventoryinfo?userId=139084

Pretty Raw Preview JSON ☰

6 - "couponTemplateInfos": [
7 -   {
8 -     "couponTemplate": {
9 -       "id": 6,
10 -      "title": "Middleware discount-2",
11 -      "summary": "Summary: Middleware course",
12 -      "desc": "Description: Middleware",
13 -      "limit": 1000,
14 -      "hasToken": true,
15 -      "background": 1,
16 -      "start": 1585638000000,
17 -      "end": 1593414000000
18 -    },
19 -     "merchants": {
20 -       "id": 6,
21 -       "name": "Middleware Shop",
22 -       "logoUrl": "https://unsplash.com/photos/Bd7gNnWJBkU",
23 -       "businessLicenseUrl": "https://www.ics.uci.edu/~cs237/",
24 -       "phone": "1234567890",
25 -       "address": "Donald Bren Hall, 6210, Irvine, CA 92697",
26 -       "isAudit": false
27 -     }
28 -   },
29 -   {
30 -     "couponTemplate": {
31 -       "id": 6,
32 -       "title": "Middleware discount-1",
33 -       "summary": "Summary: Middleware course",
34 -       "desc": "Description: Middleware",
35 -       "limit": 1000,
36 -       "hasToken": false,
37 -       "background": 1,
38 -       "start": 1582963200000,
39 -       "end": 1593414000000
40 -     },
41 -     "merchants": {
42 -       "id": 6,
```


Customer Get Coupon

```
POST 127.0.0.1:9528/coupon/gaincouponschema
Params Authorization Headers (2) Body Pre-reqs
none form-data x-www-form-urlencoded raw
1 - {
2     "userId": 139084,
3     "couponTemplate": {
4         "id": 6,
5         "title": "Middleware discount-2",
6         "hasToken": true
7     }
8 }
```

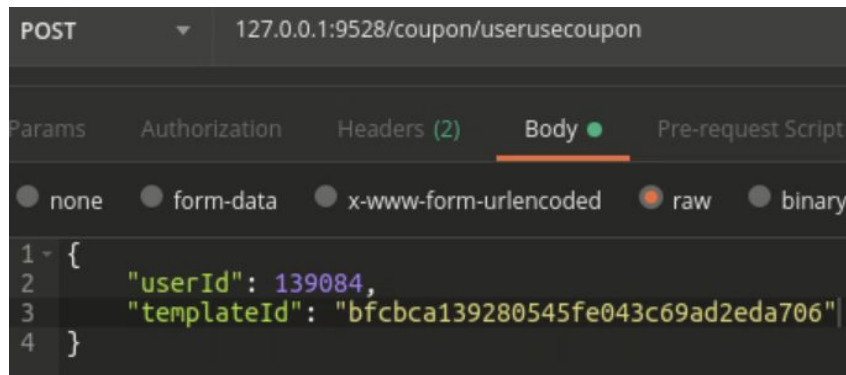
```
hbase(main):016:0> scan 'cp:coupon'
ROW COLUMN+CELL
4809319223370445078402942bfcba1392 column=i:assigned_date, timestamp=1591776372873, value=2020-06-10
80545fe043c69ad2eda706
4809319223370445078402942bfcba1392 column=i:con_date, timestamp=1591776372873, value=-1
80545fe043c69ad2eda706
4809319223370445078402942bfcba1392 column=i:template_id, timestamp=1591776372873, value=bfcba139280545fe043c69ad2eda706
80545fe043c69ad2eda706
4809319223370445078402942bfcba1392 column=i:token, timestamp=1591776372873, value=token-3
80545fe043c69ad2eda706
4809319223370445078402942bfcba1392 column=i:user_id, timestamp=1591776372873, value=\x00\x00\x00\x00\x02\x1FL
80545fe043c69ad2eda706
1 row(s) in 0.0050 seconds
```

Get Customer's Current Coupons Information

```
GET 127.0.0.1:9528/coupon/usercouponinfo/userid=139084
body Cookies Headers (3) Test Results
Pretty Raw Preview JSON
1 - {
2     "errorCode": 0,
3     "errorMsg": "",
4     "data": [
5         {
6             "coupon": {
7                 "userId": 139084,
8                 "rowKey": "4809319223370445078402942bfcba139280545fe043c69ad2eda706",
9                 "templateId": "bfcba139280545fe043c69ad2eda706",
10                "token": "token-3",
11                "assignedDate": 1578643200000,
12                "conDate": null
13            },
14            "couponTemplate": {
15                "id": 6,
16                "title": "Middleware discount-2",
17                "summary": "Summary: Middleware course",
18                "desc": "Description: Middleware",
19                "limit": 999,
20                "hasToken": true,
21                "background": 1,
22                "start": 1585638000000,
23                "end": 1593414000000
24            },
25            "merchants": {
26                "id": 6,
27                "name": "Middleware Shop",
28                "logoUrl": "https://unsplash.com/photos/Bd7gNnWJBKU",
29                "businessLicenseUrl": "https://www.lcs.ucl.edu/~cs237/",
30                "phone": "1234567890",
31                "address": "Donald Bren Hall, 6210, Irvine, CA 92697",
32                "isAudit": false
33            }
34        }
35    ]
36 }
```

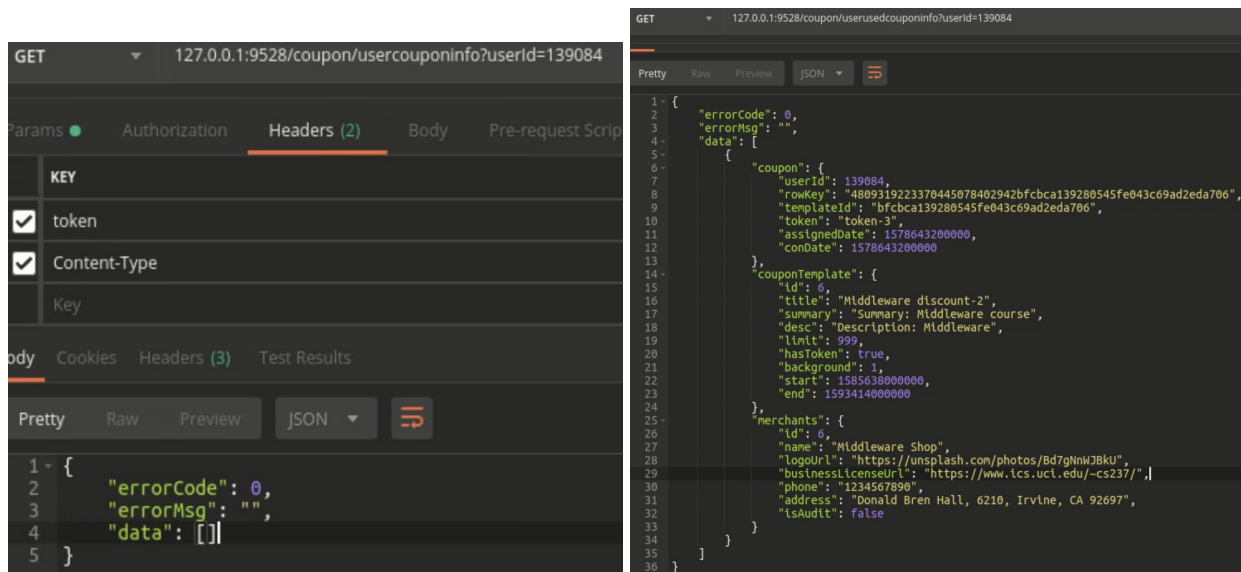
Customer Use Coupon

A Customer can use a coupon with the CouponTemplate ID



```
POST 127.0.0.1:9528/coupon/userusecoupon
Body
none form-data x-www-form-urlencoded raw binary
1 - {
2   "userId": 139084,
3   "templateId": "bfcbca139280545fe043c69ad2eda706"
4 }
```

Customer Current Coupon becomes empty and it would be available in Customer Used Coupon



```
GET 127.0.0.1:9528/coupon/usercouponinfo?userId=139084
Headers (2)
token
Content-Type
Key
body Cookies Headers (3) Test Results
Pretty Raw Preview JSON
1 - {
2   "errorCode": 0,
3   "errorMsg": "",
4   "data": []
5 }
```

```
GET 127.0.0.1:9528/coupon/userusedcouponinfo?userId=139084
Pretty Raw Preview JSON
1 - {
2   "errorCode": 0,
3   "errorMsg": "",
4   "data": [
5     {
6       "coupon": {
7         "userId": 139084,
8         "rowKey": "4809319223370445078402942bfcbca139280545fe043c69ad2eda706",
9         "templateId": "bfcbca139280545fe043c69ad2eda706",
10        "token": "token-3",
11        "assignedDate": 1578643200000,
12        "conDate": 1578643200000
13      },
14      "couponTemplate": {
15        "id": 0,
16        "title": "Middleware discount-2",
17        "summary": "Summary: Middleware course",
18        "desc": "Description: Middleware",
19        "limit": 999,
20        "hasToken": true,
21        "background": 1,
22        "start": 1585638000000,
23        "end": 1593414000000
24      },
25      "merchants": {
26        "id": 0,
27        "name": "Middleware Shop",
28        "logoUrl": "https://unsplash.com/photos/Bd7gNmJ8KU",
29        "businessLicenseUrl": "https://www.tcs.ucl.edu/~cs237/",
30        "phone": "1234567890",
31        "address": "Donald Bren Hall, 6210, Irvine, CA 92697",
32        "isAudit": false
33      }
34    }
35  ]
36 }
```

Conclusion and Future Work

For distributed systems, Kafka has been developed to collect and distribute massive messages. It is a distributed high-throughput messaging system. In Kafka, the messages are persistent and Kafka can easily scale-out. A lot of businesses have already successfully used Kafka in production (Z. Wang 2015). In this project, we implemented an integrated coupon distribution system. The functionality of the system is comprehensive and robust. Merchants are able to distribute coupons and customers can consume coupons from Kafka. All the operations will also be logged for rollback when it comes across an error. Also, all the functions are tested by Postman by HTTP request.

However, there's still something we need to do for this project. Currently, our system is deployed on a personal computer, which means all the distributed functions are implemented by pseudo-distribution. The next step is that we need to deploy it on several AWS EC2 instances to

implement the real distribution. We can make use of JMeter to do the pressure and performance test which could be used to simulate a large number of merchants and customers using the system simultaneously. We can also improve built-in monitoring service which is useful as cluster size increases. There are many mistakes that are found just as we scale. Incorporated monitoring service on sample data service can aid early detection of errors. A distributed system must be coordinated. If some event occurs, system nodes must react in an organized manner. Finally, someone has to determine how the cluster should respond and order the brokers to do something (K. Stanislav 2018). These are some of the planned changes to the current project.

Reference

Ali, A. Abdullah, M. (2018) *Recent Trends in Distributed Online Stream Processing Platform for Big Data: Survey*

Apache Kafka, <https://kafka.apache.org/>

Apache Kafka wikipedia https://en.wikipedia.org/wiki/Apache_Kafka

Dobbelaere, P. Esmaili, K. (2017) *Kafka versus RabbitMQ*

Kreps, J. Narkhede, N. Rao, J. (2011) *Kafka: a Distributed Messaging System for Log Processing*

Magnoni, L. "Modern messaging for distributed systems." Journal of Physics: Conference Series. Vol. 608. No. 1. IOP Publishing, 2015.

Noac'h, P. Costan, A. (2017) *A performance evaluation of Apache Kafka in support of big data streaming applications*

Palino, T. (2015) *Running Kafka At Scale*

Sharma, A. (2014) *Apache Kafka: Next Generation Distributed Messaging System*

Stanislav, K. (2018), "Keeping chaos at bay in the distributed world, one cluster at a time"

Z. Wang et al., "Kafka and Its Using in High-throughput and Reliable Message Distribution," 2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS), Tianjin, 2015, pp. 117-120, doi: 10.1109/ICINIS.2015.53.