Statistici Proiect EBS

Butnaru Gheorghita

1 Scurta descriere

Am implementat o arhitectura de sistem publish/subscribe, content-based, care este capabila sa proceseze publicatii si secvente de publicatii grupate in ferestre. Aplicatia este constituita din 3 module.

Primul modul numit **spg**:

1. Genereaza publicatii si subscribtii cu posibilitatea de fixare a:
   (a) numarului total de publicatii, respectiv subscriptii
   (b) ponderii pe frecventa campurilor din subscriptii si ponderii operatorilor de egalitate din subscriptii pentru campurile: city, temp
2. Publicatiile au o structura fixa de campuri
3. Implementarea contine o posibilitate de paralelizare pentru eficientizarea generarii subscriptiilor si publicatiilor (thread-uri)

Al doilea modul numit **kafka**:

1. Genereaza un stream de publicatii folosindu-se de datele generate de modulul **spg**
2. Publicatiile trec mai intai prin mecanismul de serializare binara **Google Protocol Buffers**
3. Dupa ce sunt serializate, publicatiile sunt criptate cu algoritmul de criptare simetrica **AES256**
4. Publicatiile sunt trimise la reteaua de brokeri prin intermediul unui obiect de tip **KafkaTemplate**

Al treilea modul reprezinta un **client**

1. Clientul poate salva intr-un fisier text toate subscriptiile simple si in alt fisier o singura subscriptie complexa pentru a primi notificari atunci cand acestea se potrivesc cu publicatiile
2. Clientul filtreaza publicatiile in functie de continut utilizand **Kafka Streams**, avand si posibilitatea de a procesa inclusiv ferestre de publicatii masurate in timp (ms)
3. Filtrarea publicatiilor este determinata de subscriptiile simple si subscriptiile complexe
   (a) exemplu subscriptie simpla: {(city,=,"Bucharest");(temp,$\geq$,10);(wind,<,11)} - in acest caz, clientul va accepta toate publicatiile care contin campul city = "Bucharest" cu temp $\geq$ 10 si wind < 11
   (b) exemplu sbscriptie complexa: {(city,=,"Bucharest");(avg_temp,>,8.5)} - in acest caz, clientul va accepta toate ferestrele de publicatii care au media temperaturii mai mare decat 8.5 pentru city = "Bucharest"

4. Pentru filtrarea publicatiilor, acestea sunt mai intai decriptate cu **AES256** folosindu-se cheia cu care au fost criptate, apoi sunt deserializate cu **Google Protocol Buffers**, iar dupa filtrare sunt iarasi serializate si criptate
5. In cazul in care o publicatie se potriveste cu o subscriptie simpla, publicatia respectiva este trimisa intr-un alt topic
6. In cazul in care o fereastra de publicatii se potriveste cu o subscriptie complexa, se va trimita un obiect de tip MetaPublicatie intr-un anumit topic
   (a) exemplu meta publicatie: MetaPublication{(city,=,Bucharest);(conditions,=,true);(count,=,4)} - este un mesaj special indicand ca un numar de **n** publicatii care au fost primite intr-un timp predefinit de subscriber formeaza o fereastra care are media de temperatura dorita de acelasi subscriber.
   Count = 4 reprezinta numarul de publicatii din fereastra ce contin campul city = "Bucharest"
7. Publicatiile filtrate sunt trimise in alt topic de unde pot fi preluate de alti consumatori (optional)

Pentru a porni aplicatia este nevoie de:

1. zookeeper, lasat cu configuratiile standard
2. broker1: id 1, listener pe localhost:9091
3. broker2: id 2, listener pe localhost:9092
4. broker3: id 3, listener pe localhost:9093
5. in modulul **kafka** trebuie sa se seteze urmatoarele configuratii in fisierul application.properties:
   (a) spring.kafka.bootstrap-servers=localhost:9091,localhost:9092,localhost:9093
   (b) publication.encrypt.secret-key=1234567890123456
6. in modulul **client** trebuie sa se seteze urmatoarele configuratii:
   (a) spring.kafka.bootstrap-servers=localhost:9091,localhost:9092,localhost:9093
   (b) spring.kafka.application-id.simple-subscription=kafka-streams-simple
   (c) spring.kafka.application-id.complex-subscription=kafka-streams-complex
   (d) process.subscriptions.simple=true
   (e) subscriptions.simple.input-topic=topic-ebs
   (f) subscriptions.simple.output-topic=topic-ebs-output-simple-1
   (g) process.subscriptions.complex=true
   (h) process.subscriptions.complex.window-size-millis=100
   (i) subscriptions.complex.input-topic=topic-ebs
   (j) subscriptions.complex.output-topic=topic-ebs-output-complex-1
   (k) kafka.listener.simple-input-topic.group-id=listener1
   (l) kafka.listener.simple-output-topic.group-id=listener2
   (m) kafka.listener.complex-output-topic.group-id=listener3
   (n) publication.encrypt.secret-key=1234567890123456

Fiecare topic are un numar de 3 partitii ceea ce permite o scalabilitate si o distributie eficienta a mesajelor, iar pentru fiecare partitie exista 3 replici ceea ce asigura redundanta (exista mai multe copii identice ale unei partitii în sistem) si disponibilitatea datelor (se refera la faptul ca serviciul este intotdeauna disponibil pentru utilizatori).

Aceasta arhitectura cu replici ofera rezistenta la esecuri si asigura disponibilitatea datelor, deoarece exista copii redundante ale mesajelor. Astfel, chiar dacă un nod sau replica nu funcționează corect, sistemul poate continua sa livreze si sa consume mesaje prin intermediul celorlalte noduri si/sau replici disponibile.

2 Evaluare a sistemului

2.1: Cate publicatii se livreaza cu succes prin reteaua de brokeri intr-un interval continuu de feed de 3 minute (180.000 ms). (*)

Rezultatele pentru 3 rulari:

Table 1: Results for 3 executions

Execution	Success	Failure	Time (ms)
1	96,000,000	0	181,863
2	138,000,000	0	182,157
3	132,000,000	0	181,407

Media:

Table 2: Average of the results

Success	122,000,000
Failure	0
Time (ms)	181,809

* S-a luat in calcul doar timpul consumat pentru trimiterea mesajelor (3 minute), nu si timpul consumat pentru generarea publicatiilor.

2.2: Latenta medie de livrare a unei publicatii (timpul de la emitere pana la primire) pentru publicatiile trimise in acelasi interval.

Pentru aceasta statistica am generat 20 de publicatii si am masurat timpul de la emitere pana la primire pentru fiecare publicatie. Pentru a determina latenta medie de livrare a unei publicatii, am calculat un interval de incredere de 90% folosind procedura t: $\bar{x} \pm t_{\alpha/2,n-1}\frac{s}{\sqrt{n}}$ cu urmatoarele informatii:

$$\text{latenta pentru fiecare publicatie (ms)} : 3 \times 72,\ 8 \times 74,\ 9 \times 73$$
$$n = 20,\ s = 9.75,\ \bar{x} = 73.25,\ \alpha = 0.1,\ t_{0.05,19} = 1.729$$

Intervalul de incredere de confidenta 90% pentru medie: [69.48, 77.02]

2.3: Rata de potrivire (matching) pentru cazul in care subscriptiile generate contin pe unul dintre campuri doar operator de egalitate (100%) comparata cu situatia in care frecventa operatorului de egalitate pe campul respectiv este aproximativ un sfert (25%)

```
-----------------------------------------------------------------------------------------
Detalii publicatii
-----------------------------------------------------------------------------------------
Fiecare publicatie contine 7 campuri: stationId, city, temp, rain, wind, direction, date.

Domeniul de valori pentru fiecare camp:
city: [1, 500]
stationId: [1, 10]
rain: [0.0, 100.0]
temp: [-40, 50]
direction: [1, 8]
date: [1, 12]
wind: [0, 100]

Operatorii disponibili pentru stationId, temp, rain, wind, date: =, !=, <, <=, >, >=
Operatorii disponibili pentru city, direction: =, !=

-----------------------------------------------------------------------------------------
Detalii subscriptii
-----------------------------------------------------------------------------------------

O subscriptie poate sa contina 3 campuri: city, temp, wind.

Pentru generarea subscriptiilor s-au folosit urmatoarele frecvente de aparitie pentru campuri:
city: 100%
temp: 100%
wind: 100%

Pentru generarea operatorilor de egalitate pentru frecventa minima de aparitie in campurile
"city" si "temp", s-au utilizat diferite frecvente, asa cum este prezentat in tabelul de mai jos.
In cazul campului "wind", operatorii au fost selectati complet aleatoriu dintre cei disponibili.
```

Pentru determinarea ratei de potrivire dintre subscriptii si publicatii, am folosit mai multe seturi de date generate aleator, iar rezultatele se afla in tabelul de mai jos.

sub.	pub.	frec. min. op. eg. temp (%)	frec. min. op. eg. city (%)	rata potrivire (%)
50	10000	25	0	100
50	10000	100	0	10.6
100	100000	50	50	100
100	100000	80	80	99.53
100	100000	100	100	0.12
1000	200000	0	0	100
1000	200000	50	50	100
1000	200000	80	80	100
1000	200000	100	80	52
1000	200000	100	100	2.08
10000	1000000	25	0	100
10000	1000000	100	0	100
10000	1000000	100	25	100
10000	1000000	100	80	99.72
10000	1000000	100	100	17.53