阿里P8面試題：Kafka如何做到發送端和接收端的順序一致性？

Java技術迷 2022-03-23 17:21

以下文章來源於四猿外 ， 作者四猿外

四猿外

百人技術團隊的技術總監，專注分享技術和職場心得。

點擊關注公眾號，Java乾貨及時送達 👇

Java技術迷

專注Java技術乾貨分享，Java基礎技術、數據結構、相關工具、spring Cloud、intelli...

67篇原創內容

公眾號

今天這篇文章，寫一個面試題的詳解。

為了帶著問題去學習，我特意找一個大廠朋友要了一份他們的面試題，公司名不說了，難度大概相當於P8。

面試題裡有這麼一道題：

> Kafka 如何做到發送端和接收端的順序一致性？

我在網上找了找資料、答案，結果發現很多寫的不對，或者寫的已經過時了。

於是，我決定自己寫篇文章，詳解一下這道題。

Producer 端：

Kafka 的發送端發送消息，如果是默認參數什麼都不設置，則消息如果在網絡沒有抖動的時候，可以一批批的按消息發送的順序被發送到Kafka 服務器端。但是，一旦網絡波動了，則消息就可能出現失序。

所以，要嚴格保證Kafka 發消息有序，首先要考慮同步發送消息。

同步發送消息有兩種方式：

第一種方式：設置消息響應參數 `acks > 0` ，最好是-1。

然後，設置

```
max.in.flight.requests.per.connection = 1
```

這樣設置完後，在Kafka 的發送端，將會一條消息發出後，響應必須滿足acks 設置的參數後，才會發送下一條消息。所以，雖然在使用時，還是異步發送的方式，其實底層已經是一條接一條的發送了。

第二種方式：當調用KafkaProducer 的send 方法後，調用send 方法返回的Future 對象的get 方式阻塞等待結果。等結果返回後，再繼續調用KafkaProducer 的send 方法發送下一條消息。

同步發送消息之外，還要考慮消息重發問題。

Kafka 發送端可以在發送出現問題時，判斷問題是否可以自動恢復，如果是可以自動恢復的問題，可以通過設置 `retries > 0` ，讓Kafka 自動重試。

根據Kafka 版本的不同，Kafka 1.0 之後的版本，發送端引入了冪等特性。引入冪等特性，我們可以這麼設置

```
enable.idempotence = true
```

冪等特性這個特性可以給消息添加序列號，每次發送，會把序列號遞增1。

開啟了Kafka 發送端的冪等特性後，我們就可以設置

```
max.in.flight.requests.per.connection = 5
```

這樣，當Kafka 發消息的時候，由於消息有了序列號，當發送消息出現錯誤的時候，在Kafka 底層會通過獲取服務器端的最近幾條日誌的序列號和發送端需要重新發送的消息序列號做對比，如果是連續的，那麼就可以繼續發送消息，保證消息順序。

Broker 端：

Kafka 的Topic 只是一個邏輯概念。而組成Topic 的分區才是真正存消息的地方。

Kafka 只保證同個分區內的消息是有序的。所以，如果要保證業務全局嚴格有序，就要設置Topic 為單分區的形式。

不過，往往我們的業務是不需要考慮全局有序的，我們只需要保證業務中不同類別的消息有序即可。對這些業務中不同類別的消息，可以設置成不同的Key，然後根據Key 取模。這樣，由於同類別消息有同樣的Key，就會被分配到同樣的分區中，保證有序。

但是，這裡有個問題，就是當我們對分區的數量進行改變的時候，會把以前可能分到同樣的分區的消息，分到別的分區上。這就不能保證消息順序了。

面對這種情況，就需要在動態變更分區的時候，考慮對業務的影響。有可能需要根據業務和當前分區需求，重新劃分消息類別。

另外，如果一個Topic 存在多分區的情況，並且 `min.insync.replicas` 指定的副本個數掛掉了，那麼，就會出現這種情況：發送消息寫入不了對應分區，但是消費依然可以消費消息。

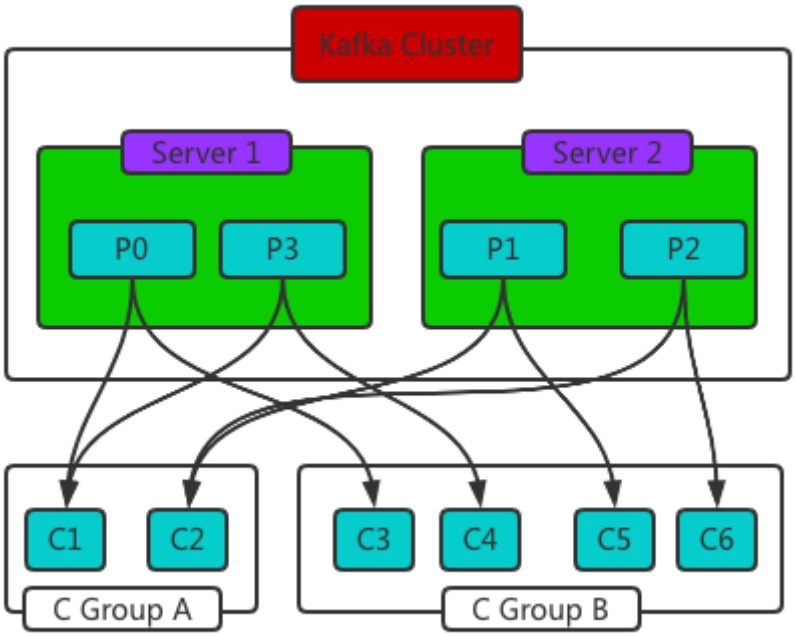
此時，往往我們會保證可用性，會考慮切換消息的分區，一旦這樣做，消息順序就可能出現不一致的情況。

所以，一定要保證 `min.insync.replicas` 參數配置的合適，去最大可能保證消息寫入的順序性。

Consumer 端：

在消費者端，根據Kafka 的模型，一個Topic 下的每個分區只能從屬於監聽這個Topic 的消費者組中的某一個消費者。

假設Topic 的分區數量為P，而消費者組中的消費者數為C。那麼，如果P < C , 就會出現消費者空閒的情況；如果P > C，則會出現一個消費者被分配多個分區的情況，如下圖。



所以，當我們消費者端使用poll 方法的時候，一定要注意：poll 方法獲取到的記錄，很可能是多個分區甚至多個Topic 的。

還需要通過ConsumerRecords 的records(TopicPartition partition) 進行進一步的排序和篩選，才能真正的保證發送和消費的順序一致性使用。

另外一個要注意的地方就是消費者的Rebalance。Rebalance 就是讓一個消費者組下所有的消費者實例，就如何消費訂閱主題的所有分區達成共識的過程。

這個Rebalance 機制是Kafka 最臭名照顧的地方：

- 它每次Rebalance，都會讓全部消費者組的消費暫停。
- 再就是Rebalance 的bug 非常多，比如就是Rebalance 後，要么某個消費者突然崩了，要么是消費者組中某些消費者停了。
- 由於Rebalance 相當於讓消費者組重新分配分區，這就可能造成消費者在Rebalance 前、後所對應的分區不一致。分區不一致，那自然消費順序就不可能一致了。

所以，我們都會盡量不讓Rebalance 發生。有三種情況會觸發Kafka 消費者的Rebalance 發生：

1. 消費者組成員發生變化：這個往往是指，我們認為增減了組內的消費者個數，又或者是某些消費者崩潰了，導致被踢出組。
2. 訂閱主題數發生變化：Kafka 的消費者組是能用正則去模糊匹配Topic 的。這就造成一個問題，當我們在Kafka 中添加主題後，可能會造成消費者組監聽的Topic 數發生變化。
3. 訂閱主題的分區數發生變化：有些時候，可能我們想動態的線上變更主題的分區數。

所以，當這三種情況觸發Rebalance 後，就會出現問題，消費順序不一致只是其中很輕微的一種負面影響。

寫到這裡，基本算是徹底把這道面試題回答完整。

其中Producer 端的冪等性質，Consumer 端的Rebalance 情況，是最容易在回答Kafka 順序一致性類似面試題中漏掉的。而很多網上的面試題答案都已經相對過時了，沒有談過這兩個特性相關的問題。

所以，大家做面試準備的時候，一定要好好的複習相關中間件的知識和特性，而不是去死記硬背答案。希望大家學習的時候多加註意。

往 期 推 薦

1、Windows新功能太“社死”！教你一鍵快速禁用

2、發現競爭對手代碼中的低級Bug後，我被公司解僱並送上了法庭

3、為什麼說技術人一定要有產品思維

4、操作系統聯合創始人反目成仇，這個Linux發行版危在旦夕

5、Java8八年不倒、IntelliJ IDEA力壓Eclipse