Key为null时Kafka如何选择分区(Partition)

我们往Kafka发送消息时一般都是将消息封装到KeyedMessage类中：

```
val message = new KeyedMessage[String, String](topic, key, content)
producer.send(message)
```

Kafka会根据传进来的key计算其分区ID。但是这个Key可以不传，根据Kafka的官方文档描述：如果key为null，那么Producer将会把这条消息发送给随机的一个Partition。

> If the key is null, then the Producer will assign the message to a random Partition.

这句话从字面上理解是每条消息只要没有设置key(null)，那么这条消息就会随机发送给一个Partition。但是代码实现是这么做的么(肯定不是，否则就没有这篇文章)？我们来看看Kafka是如何计算Partition ID的：

```
private def getPartition(topic: String, key: Any,
    topicPartitionList: Seq[PartitionAndLeader]): Int = {
 val numPartitions = topicPartitionList.size
 if(numPartitions <= 0)
  throw new UnknownTopicOrPartitionException("Topic " + topic + " doesn't exist")
 val partition =
  if(key == null) {
   // If the key is null, we don't really need a partitioner
   // So we look up in the send partition cache for the
   // topic to decide the target partition
   val id = sendPartitionPerTopicCache.get(topic)
   id match {
    case Some(partitionId) =>
     // directly return the partitionId without checking availability
     // of the leader, since we want to postpone the failure until
     // the send operation anyways
     partitionId
    case None =>
     val availablePartitions = topicPartitionList
           .filter(_.leaderBrokerIdOpt.isDefined)
     if (availablePartitions.isEmpty)
      throw new LeaderNotAvailableException("No
        leader for any partition in topic " + topic)
```

```
        val index = Utils.abs(Random.nextInt) % availablePartitions.size
        val partitionId = availablePartitions(index).partitionId
        sendPartitionPerTopicCache.put(topic, partitionId)
        partitionId
      }
    } else
      partitioner.partition(key, numPartitions)
  if(partition < 0 || partition >= numPartitions)
    throw new UnknownTopicOrPartitionException("Invalid partition id: "
       + partition + " for topic " + topic
       + "; Valid values are in the inclusive range of [0, "
       + (numPartitions-1) + "]")
  trace("Assigning message of topic %s and key %s to a selected partition %d
  ".format(topic, if (key == null) "[none]" else key.toString, partition))
  partition
}
```

从上面的代码可以看出，如果key == null,则从sendPartitionPerTopicCache(sendPartitionPerTopicCache的类型是HashMap.empty[String, Int])中获取分区ID，如果找到了就直接用这个分区ID；否则随机去选择一个partitionId，并将partitionId存放到sendPartitionPerTopicCache中去。而且sendPartitionPerTopicCache是每隔topic.metadata.refresh.interval.ms（这个参数是不是很熟悉？昨天在《Kafka Producer是如何动态感知Topic分区数变化》文章中首次介绍了这个参数）时间才会清空的：

```
if (topicMetadataRefreshInterval >= 0 &&
      SystemTime.milliseconds - lastTopicMetadataRefreshTime >
      topicMetadataRefreshInterval) {
  Utils.swallowError(brokerPartitionInfo
  .updateInfo(topicMetadataToRefresh.toSet, correlationId.getAndIncrement))
  sendPartitionPerTopicCache.clear()
  topicMetadataToRefresh.clear
  lastTopicMetadataRefreshTime = SystemTime.milliseconds
}
```

也就是说在key为null的情况下，Kafka并不是每条消息都随机选择一个Partition；而是每隔topic.metadata.refresh.interval.ms才会随机选择一次！别被文档所骗啊！

不过LinkedIn工程师Guozhang Wang解释到：本来producer在key为null的情况下每条消息都随机选择一个Partition，但后面改成这种伪随机的以此来减少服务器端的sockets数。

> Originally the producer behavior under null-key is "random" random, but later changed to this "periodic" random to reduce the number of sockets on the server side: imagine if you have n brokers and m producers where m >>> n, with random random distribution each server will need to maintain a socket with each of the m producers.
>
> We realized that this change IS misleading and we have changed back to random random in the new producer released in 0.8.2.

在Kafka new producer上如果Key为null则每条消息都会选择不同的Partition：

```
if (record.partition() != null) {
    // they have given us a partition, use it
    if (record.partition() < 0 || record.partition() >= numPartitions)
        throw new IllegalArgumentException("Invalid partition given with record: "
                                           + record.partition()
                                           + " is not in the range [0..."
                                           + numPartitions
                                           + "].");
    return record.partition();
} else if (record.key() == null) {
    int nextValue = counter.getAndIncrement();
    List<PartitionInfo> availablePartitions = cluster
        .availablePartitionsForTopic(record.topic());
    if (availablePartitions.size() > 0) {
        int part = Utils.abs(nextValue) % availablePartitions.size();
        return availablePartitions.get(part).partition();
    } else {
        // no partitions are available, give a non-available partition
        return Utils.abs(nextValue) % numPartitions;
    }
} else {
    // hash the key to choose a partition
    return Utils.abs(Utils.murmur2(record.key())) % numPartitions;
}
```

可以看出这是一种round-robin模式选择分区ID的。