Kafka consumer元数据metadata更新

在new KafkaConsumer的时候，会新建Metadata。

ClusterResourceListeners clusterResourceListeners = configureClusterResourceListeners(keyDeserializer, valueDeserializer, reporters, interceptorList);  
this.metadata = new Metadata(retryBackoffMs, config.getLong(ConsumerConfig.*METADATA\_MAX\_AGE\_CONFIG*),  
 true, false, clusterResourceListeners);  
List<InetSocketAddress> addresses = ClientUtils.*parseAndValidateAddresses*(config.getList(ConsumerConfig.*BOOTSTRAP\_SERVERS\_CONFIG*));  
this.metadata.update(Cluster.*bootstrap*(addresses), Collections.<String>*emptySet*(), 0);

retryBackoffMs 重试补偿毫秒数

*METADATA\_MAX\_AGE\_CONFIG* meta数据配置的age时长

Cluster.*bootstrap*(addresses) 设置配置的服务器地址

修改cluster的meta信息

*/\*\*  
 \* Updates the cluster metadata. If topic expiry is enabled, expiry time  
 \* is set for topics if required and expired topics are removed from the metadata.  
 \*  
 \** ***@param*** *newCluster the cluster containing metadata for topics with valid metadata  
 \** ***@param*** *unavailableTopics topics which are non-existent or have one or more partitions whose  
 \* leader is not known  
 \** ***@param*** *now current time in milliseconds  
 \*/*public synchronized void update(Cluster newCluster, Set<String> unavailableTopics, long now) {  
 Objects.*requireNonNull*(newCluster, "cluster should not be null");  
  
 this.needUpdate = false;  
 this.lastRefreshMs = now;  
 this.lastSuccessfulRefreshMs = now;  
 this.version += 1;  
  
 if (topicExpiryEnabled) {  
 // Handle expiry of topics from the metadata refresh set.  
 for (Iterator<Map.Entry<String, Long>> it = topics.entrySet().iterator(); it.hasNext(); ) {  
 Map.Entry<String, Long> entry = it.next();  
 long expireMs = entry.getValue();  
 if (expireMs == *TOPIC\_EXPIRY\_NEEDS\_UPDATE*)  
 entry.setValue(now + *TOPIC\_EXPIRY\_MS*);  
 else if (expireMs <= now) {  
 it.remove();  
 *log*.debug("Removing unused topic {} from the metadata list, expiryMs {} now {}", entry.getKey(), expireMs, now);  
 }  
 }  
 }  
  
 for (Listener listener: listeners)  
 listener.onMetadataUpdate(newCluster, unavailableTopics);  
  
 String previousClusterId = cluster.clusterResource().clusterId();  
  
 if (this.needMetadataForAllTopics) {  
 // the listener may change the interested topics, which could cause another metadata refresh.  
 // If we have already fetched all topics, however, another fetch should be unnecessary.  
 this.needUpdate = false;  
 this.cluster = getClusterForCurrentTopics(newCluster);  
 } else {  
 this.cluster = newCluster;  
 }  
  
 // The bootstrap cluster is guaranteed not to have any useful information  
 if (!newCluster.isBootstrapConfigured()) {  
 String newClusterId = newCluster.clusterResource().clusterId();  
 if (newClusterId == null ? previousClusterId != null : !newClusterId.equals(previousClusterId))  
 *log*.info("Cluster ID: {}", newClusterId);  
 clusterResourceListeners.onUpdate(newCluster.clusterResource());  
 }  
  
 notifyAll();  
 *log*.debug("Updated cluster metadata version {} to {}", this.version, this.cluster);  
}

这个update的功能

处理topic ，如果topic过期需要更新，过期时间增加5分钟，过期的topic移除。

刚开始这里的listener是空的。功能主要是处理不可用的topic

更新KafkaConsumer中的cluster

然后唤醒所有等待的线程。

然后new NetworkClient ConsumerNetworkClient ConsumerCoordinator Fetcher

在ConsumerCoordinator 初始化的时候回给Meta添加Listener。

Listener的功能

更新订阅的主题

在什么时候回调用NetWorkClient.java

@Override  
public void handleCompletedMetadataResponse(RequestHeader requestHeader, long now, MetadataResponse response) {  
 this.metadataFetchInProgress = false;  
 Cluster cluster = response.cluster();  
 // check if any topics metadata failed to get updated  
 Map<String, Errors> errors = response.errors();  
 if (!errors.isEmpty())  
 log.warn("Error while fetching metadata with correlation id {} : {}", requestHeader.correlationId(), errors);  
  
 // don't update the cluster if there are no valid nodes...the topic we want may still be in the process of being  
 // created which means we will get errors and no nodes until it exists  
 if (cluster.nodes().size() > 0) {  
 this.metadata.update(cluster, response.unavailableTopics(), now);  
 } else {  
 log.trace("Ignoring empty metadata response with correlation id {}.", requestHeader.correlationId());  
 this.metadata.failedUpdate(now, null);  
 }  
}

*/\*\*  
 \* Handle any completed receives and update the response list with the responses received.  
 \*  
 \** ***@param*** *responses The list of responses to update  
 \** ***@param*** *now The current time  
 \*/*private void handleCompletedReceives(List<ClientResponse> responses, long now) {  
 for (NetworkReceive receive : this.selector.completedReceives()) {  
 String source = receive.source();  
 InFlightRequest req = inFlightRequests.completeNext(source);  
 Struct responseStruct = *parseStructMaybeUpdateThrottleTimeMetrics*(receive.payload(), req.header,  
 throttleTimeSensor, now);  
 if (log.isTraceEnabled()) {  
 log.trace("Completed receive from node {} for {} with correlation id {}, received {}", req.destination,  
 req.header.apiKey(), req.header.correlationId(), responseStruct);  
 }  
 AbstractResponse body = AbstractResponse.*parseResponse*(req.header.apiKey(), responseStruct);  
 if (req.isInternalRequest && body instanceof MetadataResponse)  
 metadataUpdater.handleCompletedMetadataResponse(req.header, now, (MetadataResponse) body);  
 else if (req.isInternalRequest && body instanceof ApiVersionsResponse)  
 handleApiVersionsResponse(responses, req, now, (ApiVersionsResponse) body);  
 else  
 responses.add(req.completed(body, now));  
 }  
}

最终是定时任务Sender.run() 这个是producer那边的定时任务

Consumer是在poll的时候回调用

ConsumerCoordinator.java

private void addMetadataListener() {  
 this.metadata.addListener(new Metadata.Listener() {  
 @Override  
 public void onMetadataUpdate(Cluster cluster, Set<String> unavailableTopics) {  
 // if we encounter any unauthorized topics, raise an exception to the user  
 if (!cluster.unauthorizedTopics().isEmpty())  
 throw new TopicAuthorizationException(new HashSet<>(cluster.unauthorizedTopics()));  
  
 if (subscriptions.hasPatternSubscription())  
 updatePatternSubscription(cluster);  
  
 // check if there are any changes to the metadata which should trigger a rebalance  
 if (subscriptions.partitionsAutoAssigned()) {  
 MetadataSnapshot snapshot = new MetadataSnapshot(subscriptions, cluster);  
 if (!snapshot.equals(metadataSnapshot))  
 metadataSnapshot = snapshot;  
 }  
  
 if (!Collections.*disjoint*(metadata.topics(), unavailableTopics))  
 metadata.requestUpdate();  
 }  
 });  
}

MetaData内部接口

*/\*\*  
 \* MetadataUpdate Listener  
 \*/*public interface Listener {  
 */\*\*  
 \* Callback invoked on metadata update.  
 \*  
 \** ***@param*** *cluster the cluster containing metadata for topics with valid metadata  
 \** ***@param*** *unavailableTopics topics which are non-existent or have one or more partitions whose  
 \* leader is not known  
 \*/* void onMetadataUpdate(Cluster cluster, Set<String> unavailableTopics);  
}

获取数据

都是通过Fetcher来实现的。

// if data is available already, return it immediately  
Map<TopicPartition, List<ConsumerRecord<K, V>>> records = fetcher.fetchedRecords();

Kafka返回的数据放在下面的数据结构中

private final ConcurrentLinkedQueue<CompletedFetch> completedFetches;

这个类是读取数据的。

KafkaLZ4BlockInputStream