package com.twitter.search.earlybird.partition;

import com.google.common.base.Preconditions;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.search.common.metrics.SearchCounter;

import com.twitter.search.common.metrics.SearchLongGauge;

/\*\*

\* Keeps track of an up-to-date PartitionConfig. The PartitionConfig may be periodically reloaded

\* from ZooKeeper. If you need a consistent view of the current partition configuration, make sure

\* to grab a reference to a single PartitionConfig using getCurrentPartitionConfig() and reuse that

\* object.

\*/

public class DynamicPartitionConfig {

private static final Logger LOG = LoggerFactory.getLogger(DynamicPartitionConfig.class);

private static final SearchCounter FAILED\_UPDATE\_COUNTER\_NAME =

SearchCounter.export("dynamic\_partition\_config\_failed\_update");

private static final SearchCounter SUCCESSFUL\_UPDATE\_COUNTER =

SearchCounter.export("dynamic\_partition\_config\_successful\_update");

// We assume that DynamicPartitionConfig is practically a singleton in Earlybird app.

private static final SearchLongGauge NUM\_REPLICAS\_IN\_HASH\_PARTITION =

SearchLongGauge.export("dynamic\_partition\_config\_num\_replicas\_in\_hash\_partition");

private final PartitionConfig curPartitionConfig;

public DynamicPartitionConfig(PartitionConfig initialConfig) {

this.curPartitionConfig = initialConfig;

NUM\_REPLICAS\_IN\_HASH\_PARTITION.set(initialConfig.getNumReplicasInHashPartition());

}

public PartitionConfig getCurrentPartitionConfig() {

return curPartitionConfig;

}

/\*\*

\* Verifies that the new partition config is compatible with the old one, and if it is, updates

\* the number of replicas per partition based on the new partition config.

\*/

public void setCurrentPartitionConfig(PartitionConfig partitionConfig) {

Preconditions.checkNotNull(partitionConfig);

// For now, we only allow the number of replicas in this partition to be dynamically updated.

// Ensure that the only things that have changed between the previous

if (curPartitionConfig.getClusterName().equals(partitionConfig.getClusterName())

&& (curPartitionConfig.getMaxEnabledLocalSegments()

== partitionConfig.getMaxEnabledLocalSegments())

&& (curPartitionConfig.getNumPartitions() == partitionConfig.getNumPartitions())

&& (curPartitionConfig.getTierStartDate().equals(partitionConfig.getTierStartDate()))

&& (curPartitionConfig.getTierEndDate().equals(partitionConfig.getTierEndDate()))

&& (curPartitionConfig.getTierName().equals(partitionConfig.getTierName()))) {

if (curPartitionConfig.getNumReplicasInHashPartition()

!= partitionConfig.getNumReplicasInHashPartition()) {

SUCCESSFUL\_UPDATE\_COUNTER.increment();

curPartitionConfig.setNumReplicasInHashPartition(

partitionConfig.getNumReplicasInHashPartition());

NUM\_REPLICAS\_IN\_HASH\_PARTITION.set(partitionConfig.getNumReplicasInHashPartition());

}

} else {

FAILED\_UPDATE\_COUNTER\_NAME.increment();

LOG.warn(

"Attempted to update partition config with inconsistent layout.\n"

+ "Current: " + curPartitionConfig.getPartitionConfigDescription() + "\n"

+ "New: " + partitionConfig.getPartitionConfigDescription());

}

}

}