package com.twitter.search.earlybird.config;

import java.util.Comparator;

import java.util.SortedSet;

import com.google.common.base.Preconditions;

public final class TierInfoUtil {

public static final Comparator<TierInfo> TIER\_COMPARATOR = (t1, t2) -> {

// Reverse sort order based on date.

return t2.getDataStartDate().compareTo(t1.getDataStartDate());

};

private TierInfoUtil() {

}

/\*\*

\* Checks that the serving ranges and the override serving ranges of the given tiers do not

\* overlap, and do not have gaps. Dark reads tiers are ignored.

\*/

public static void checkTierServingRanges(SortedSet<TierInfo> tierInfos) {

boolean tierServingRangesOverlap = false;

boolean tierOverrideServingRangesOverlap = false;

boolean tierServingRangesHaveGaps = false;

boolean tierOverrideServingRangesHaveGaps = false;

TierInfoWrapper previousTierInfoWrapper = null;

TierInfoWrapper previousOverrideTierInfoWrapper = null;

for (TierInfo tierInfo : tierInfos) {

TierInfoWrapper tierInfoWrapper = new TierInfoWrapper(tierInfo, false);

TierInfoWrapper overrideTierInfoWrapper = new TierInfoWrapper(tierInfo, true);

// Check only the tiers to which we send light reads.

if (!tierInfoWrapper.isDarkRead()) {

if (previousTierInfoWrapper != null) {

if (TierInfoWrapper.servingRangesOverlap(previousTierInfoWrapper, tierInfoWrapper)) {

// In case of rebalancing, we may have an overlap data range while

// overriding with a good serving range.

if (previousOverrideTierInfoWrapper == null

|| TierInfoWrapper.servingRangesOverlap(

previousOverrideTierInfoWrapper, overrideTierInfoWrapper)) {

tierServingRangesOverlap = true;

}

}

if (TierInfoWrapper.servingRangesHaveGap(previousTierInfoWrapper, tierInfoWrapper)) {

tierServingRangesHaveGaps = true;

}

}

previousTierInfoWrapper = tierInfoWrapper;

}

if (!overrideTierInfoWrapper.isDarkRead()) {

if (previousOverrideTierInfoWrapper != null) {

if (TierInfoWrapper.servingRangesOverlap(previousOverrideTierInfoWrapper,

overrideTierInfoWrapper)) {

tierOverrideServingRangesOverlap = true;

}

if (TierInfoWrapper.servingRangesHaveGap(previousOverrideTierInfoWrapper,

overrideTierInfoWrapper)) {

tierOverrideServingRangesHaveGaps = true;

}

}

previousOverrideTierInfoWrapper = overrideTierInfoWrapper;

}

}

Preconditions.checkState(!tierServingRangesOverlap,

"Serving ranges of light reads tiers must not overlap.");

Preconditions.checkState(!tierServingRangesHaveGaps,

"Serving ranges of light reads tiers must not have gaps.");

Preconditions.checkState(!tierOverrideServingRangesOverlap,

"Override serving ranges of light reads tiers must not overlap.");

Preconditions.checkState(!tierOverrideServingRangesHaveGaps,

"Override serving ranges of light reads tiers must not have gaps.");

}

}