package com.twitter.search.earlybird;

import com.google.common.annotations.VisibleForTesting;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.common.util.Clock;

import com.twitter.common.zookeeper.ServerSet;

import com.twitter.decider.Decider;

import com.twitter.search.common.decider.DeciderUtil;

import com.twitter.search.earlybird.partition.PartitionConfig;

import com.twitter.search.earlybird.partition.SearchIndexingMetricSet;

import com.twitter.search.earlybird.thrift.EarlybirdStatusCode;

public class EarlybirdWarmUpManager {

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

private static final String WARM\_UP\_ON\_DURATION\_DECIDER\_KEY\_PATTERN =

"%s\_warm\_up\_duration\_seconds";

private final EarlybirdServerSetManager earlybirdServerSetManager;

private final String clusterName;

private final SearchIndexingMetricSet.StartupMetric startUpInWarmUpMetric;

private final Decider decider;

private final Clock clock;

public EarlybirdWarmUpManager(EarlybirdServerSetManager earlybirdServerSetManager,

PartitionConfig partitionConfig,

SearchIndexingMetricSet searchIndexingMetricSet,

Decider decider,

Clock clock) {

this.earlybirdServerSetManager = earlybirdServerSetManager;

this.clusterName = partitionConfig.getClusterName();

this.startUpInWarmUpMetric = searchIndexingMetricSet.startupInWarmUp;

this.decider = decider;

this.clock = clock;

}

public String getServerSetIdentifier() {

return earlybirdServerSetManager.getServerSetIdentifier();

}

/\*\*

\* Warms up the earlybird. The earlybird joins a special server set that gets production dark

\* reads, and leaves this server set after a specified period of time.

\*/

public void warmUp() throws InterruptedException, ServerSet.UpdateException {

int warmUpDurationSeconds = DeciderUtil.getAvailability(

decider,

String.format(WARM\_UP\_ON\_DURATION\_DECIDER\_KEY\_PATTERN, clusterName.replaceAll("-", "\_")));

if (warmUpDurationSeconds == 0) {

LOG.info(String.format("Warm up stage duration for cluster %s set to 0. Skipping.",

clusterName));

return;

}

earlybirdServerSetManager.joinServerSet("internal warm up");

// If doWarmUp() is interrupted, try to leave the server set, and propagate the

// InterruptedException. Otherwise, try to leave the server set, and propagate any exception

// that it might throw.

InterruptedException warmUpInterruptedException = null;

try {

doWarmUp(warmUpDurationSeconds);

} catch (InterruptedException e) {

warmUpInterruptedException = e;

throw e;

} finally {

if (warmUpInterruptedException != null) {

try {

earlybirdServerSetManager.leaveServerSet("internal warm up");

} catch (Exception e) {

warmUpInterruptedException.addSuppressed(e);

}

} else {

earlybirdServerSetManager.leaveServerSet("internal warm up");

}

}

}

@VisibleForTesting

protected void doWarmUp(int warmUpDurationSeconds) throws InterruptedException {

long warmUpStartTimeMillis = clock.nowMillis();

LOG.info(String.format("Warming up for %d seconds.", warmUpDurationSeconds));

EarlybirdStatus.beginEvent("warm\_up", startUpInWarmUpMetric);

// Sleep for warmUpDurationSeconds seconds, but check if the server is going down every second.

int count = 0;

try {

while ((count++ < warmUpDurationSeconds)

&& (EarlybirdStatus.getStatusCode() != EarlybirdStatusCode.STOPPING)) {

clock.waitFor(1000);

}

} finally {

LOG.info(String.format("Done warming up after %d milliseconds.",

clock.nowMillis() - warmUpStartTimeMillis));

EarlybirdStatus.endEvent("warm\_up", startUpInWarmUpMetric);

}

}

}