package com.twitter.search.earlybird.partition;

import com.twitter.common.base.Supplier;

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

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

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

/\*\*

\* Exporting per-segment stats collected in {@link SegmentIndexStats}.

\*

\* This class tries to reuse stat prefixes of "segment\_stats\_[0-N]\_\*" where N is the number

\* of segments managed by this earlybird.

\* For example, stats prefixed with "segment\_stats\_0\_\*" always represent the most recent segment.

\* As we add more segments (and drop older ones), the same "segment\_stats\_\*" stats end up exporting

\* data for different underlying segments.

\*

\* This is done as an alternative to exporting stats that have the timesliceId in them, which

\* would avoid the need for reusing the same stat names, but would create an ever-increasing set

\* of unique stats exported by earlybirds.

\*/

public final class SegmentIndexStatsExporter {

private static final class StatReader extends SearchMetric<Long> {

private volatile Supplier<Number> counter = () -> 0;

private StatReader(String name) {

super(name);

}

@Override

public Long read() {

return counter.get().longValue();

}

@Override

public void reset() {

counter = () -> 0;

}

}

private SegmentIndexStatsExporter() {

}

private static final String NAME\_PREFIX = "segment\_stats\_";

/\*\*

\* Exports stats for some counts for the given segment:

\* - status\_count: number of tweets indexed

\* - delete\_count: number of deletes indexed

\* - partial\_update\_count: number of partial updates indexed

\* - out\_of\_order\_update\_count: number of out of order updates indexed

\* - segment\_size\_bytes: the segment size in bytes

\*

\* @param segmentInfo The segment for which these stats should be exported.

\* @param segmentIndex The index of this segment in the list of all segments.

\*/

public static void export(SegmentInfo segmentInfo, int segmentIndex) {

exportStat(segmentIndex, "status\_count",

() -> segmentInfo.getIndexStats().getStatusCount());

exportStat(segmentIndex, "delete\_count",

() -> segmentInfo.getIndexStats().getDeleteCount());

exportStat(segmentIndex, "partial\_update\_count",

() -> segmentInfo.getIndexStats().getPartialUpdateCount());

exportStat(segmentIndex, "out\_of\_order\_update\_count",

() -> segmentInfo.getIndexStats().getOutOfOrderUpdateCount());

exportStat(segmentIndex, "segment\_size\_bytes",

() -> segmentInfo.getIndexStats().getIndexSizeOnDiskInBytes());

SearchLongGauge timeSliceIdStat =

SearchLongGauge.export(NAME\_PREFIX + segmentIndex + "\_timeslice\_id");

timeSliceIdStat.set(segmentInfo.getTimeSliceID());

}

private static void exportStat(final int segmentIndex,

final String nameSuffix,

Supplier<Number> counter) {

final String name = getName(segmentIndex, nameSuffix);

StatReader statReader = SearchMetricsRegistry.registerOrGet(

() -> new StatReader(name), name, StatReader.class);

statReader.counter = counter;

}

private static String getName(final int segmentIndex, final String nameSuffix) {

return NAME\_PREFIX + segmentIndex + "\_" + nameSuffix;

}

}