package com.twitter.search.earlybird.archive.segmentbuilder;

import java.util.Collection;

import com.google.common.collect.ImmutableList;

import com.google.inject.Module;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.app.Flaggable;

import com.twitter.inject.server.AbstractTwitterServer;

import com.twitter.util.Future;

import com.twitter.util.Time;

public class SegmentBuilderApp extends AbstractTwitterServer {

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

public SegmentBuilderApp() {

createFlag("onlyRunOnce",

true,

"whether to stop segment builder after one loop",

Flaggable.ofBoolean());

createFlag("waitBetweenLoopsMins",

60,

"how many minutes to wait between building loops",

Flaggable.ofInt());

createFlag("startup\_batch\_size",

30,

"How many instances can start and read timeslice info from HDFS at the same time. "

+ "If you don't know what this parameter is, please do not change this parameter.",

Flaggable.ofInt());

createFlag("instance",

20,

"the job instance number",

Flaggable.ofInt());

createFlag("segmentZkLockExpirationHours",

0,

"max hours to hold the zookeeper lock while building segment",

Flaggable.ofInt());

createFlag("startupSleepMins",

2L,

"sleep multiplier of startupSleepMins before job runs",

Flaggable.ofLong());

createFlag("maxRetriesOnFailure",

3,

"how many times we should try to rebuild a segment when failure happens",

Flaggable.ofInt());

createFlag("hash\_partitions",

ImmutableList.of(),

"comma separated hash partition ids, e.g., 0,1,3,4. "

+ "If not specified, all the partitions will be built.",

Flaggable.ofJavaList(Flaggable.ofInt()));

createFlag("numSegmentBuilderPartitions",

100,

"Number of partitions for dividing up all segment builder work",

Flaggable.ofInt());

createFlag("waitBetweenSegmentsSecs",

10,

"Time to sleep between processing segments.",

Flaggable.ofInt());

createFlag("waitBeforeQuitMins",

2,

"How many minutes to sleep before quitting.",

Flaggable.ofInt());

createFlag("scrubGen",

"",

"Scrub gen for which segment builders should be run.",

Flaggable.ofString());

}

@Override

public void start() {

SegmentBuilder segmentBuilder = injector().instance(SegmentBuilder.class);

closeOnExit((Time time) -> {

segmentBuilder.doShutdown();

return Future.Unit();

});

LOG.info("Starting run()");

segmentBuilder.run();

LOG.info("run() complete");

// Now shutdown

shutdown();

}

protected void shutdown() {

LOG.info("Calling close() to initiate shutdown");

close();

}

@Override

public Collection<Module> javaModules() {

return ImmutableList.of(new SegmentBuilderModule());

}

}