package com.twitter.search.earlybird\_root;

import java.util.List;

import java.util.concurrent.TimeUnit;

import javax.annotation.Nullable;

import javax.inject.Named;

import javax.inject.Singleton;

import com.google.inject.Key;

import com.google.inject.Provides;

import com.twitter.app.Flag;

import com.twitter.app.Flaggable;

import com.twitter.common.util.Clock;

import com.twitter.finagle.Service;

import com.twitter.finagle.memcached.JavaClient;

import com.twitter.finagle.stats.StatsReceiver;

import com.twitter.inject.TwitterModule;

import com.twitter.search.common.caching.Cache;

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

import com.twitter.search.common.root.LoggingSupport;

import com.twitter.search.common.root.PartitionConfig;

import com.twitter.search.common.root.PartitionLoggingSupport;

import com.twitter.search.common.root.RootClientServiceBuilder;

import com.twitter.search.common.root.SearchRootModule;

import com.twitter.search.common.root.SearchRootWarmup;

import com.twitter.search.common.root.SplitterService;

import com.twitter.search.common.root.ValidationBehavior;

import com.twitter.search.common.root.WarmupConfig;

import com.twitter.search.common.schema.earlybird.EarlybirdCluster;

import com.twitter.search.earlybird.config.TierInfoSource;

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

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

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

import com.twitter.search.earlybird\_root.caching.DefaultForcedCacheMissDecider;

import com.twitter.search.earlybird\_root.caching.RecencyCache;

import com.twitter.search.earlybird\_root.caching.RelevanceCache;

import com.twitter.search.earlybird\_root.caching.StrictRecencyCache;

import com.twitter.search.earlybird\_root.caching.TermStatsCache;

import com.twitter.search.earlybird\_root.caching.TopTweetsCache;

import com.twitter.search.earlybird\_root.caching.TopTweetsServicePostProcessor;

import com.twitter.search.earlybird\_root.common.EarlybirdRequestContext;

import com.twitter.search.earlybird\_root.filters.RequestContextToEarlybirdRequestFilter;

import com.twitter.util.Future;

import static com.twitter.search.earlybird\_root.EarlybirdCommonModule.NAMED\_ALT\_CLIENT;

public class FullArchiveRootModule extends TwitterModule {

private static final String CLUSTER = "archive\_full";

private static final String ALT\_TRAFFIC\_PERCENTAGE\_DECIDER\_KEY =

"full\_archive\_alt\_client\_traffic\_percentage";

private final Flag<Boolean> forceAltClientFlag = createFlag(

"force\_alt\_client",

false,

"Always sends traffic to the alt client",

Flaggable.ofJavaBoolean());

@Override

public void configure() {

bind(Key.get(EarlybirdCluster.class)).toInstance(EarlybirdCluster.FULL\_ARCHIVE);

bind(EarlybirdServiceScatterGatherSupport.class)

.to(EarlybirdFullArchiveScatterGatherSupport.class);

bind(EarlybirdService.ServiceIface.class).to(FullArchiveRootService.class);

}

@Provides

LoggingSupport<EarlybirdRequest, EarlybirdResponse> provideLoggingSupport(

SearchDecider decider) {

return new EarlybirdServiceLoggingSupport(decider);

}

@Provides

PartitionLoggingSupport<EarlybirdRequestContext> providePartitionLoggingSupport() {

return new EarlybirdServicePartitionLoggingSupport();

}

@Provides

ValidationBehavior<EarlybirdRequest, EarlybirdResponse> provideValidationBehavior() {

return new EarlybirdServiceValidationBehavior();

}

@Provides

@Singleton

@Nullable

@Named(NAMED\_ALT\_CLIENT)

EarlybirdServiceChainBuilder provideAltEarlybirdServiceChainBuilder(

@Named(NAMED\_ALT\_CLIENT) @Nullable PartitionConfig altPartitionConfig,

RequestContextToEarlybirdRequestFilter requestContextToEarlybirdRequestFilter,

EarlybirdTierThrottleDeciders tierThrottleDeciders,

@Named(SearchRootModule.NAMED\_NORMALIZED\_SEARCH\_ROOT\_NAME) String normalizedSearchRootName,

SearchDecider decider,

TierInfoSource tierConfig,

@Named(NAMED\_ALT\_CLIENT) @Nullable

RootClientServiceBuilder<EarlybirdService.ServiceIface> altRootClientServiceBuilder,

PartitionAccessController partitionAccessController,

StatsReceiver statsReceiver

) {

if (altPartitionConfig == null || altRootClientServiceBuilder == null) {

return null;

}

return new EarlybirdServiceChainBuilder(

altPartitionConfig,

requestContextToEarlybirdRequestFilter,

tierThrottleDeciders,

normalizedSearchRootName,

decider,

tierConfig,

altRootClientServiceBuilder,

partitionAccessController,

statsReceiver

);

}

@Provides

@Singleton

@Nullable

@Named(NAMED\_ALT\_CLIENT)

EarlybirdChainedScatterGatherService provideAltEarlybirdChainedScatterGatherService(

@Named(NAMED\_ALT\_CLIENT) @Nullable EarlybirdServiceChainBuilder altServiceChainBuilder,

EarlybirdServiceScatterGatherSupport scatterGatherSupport,

PartitionLoggingSupport<EarlybirdRequestContext> partitionLoggingSupport

) {

if (altServiceChainBuilder == null) {

return null;

}

return new EarlybirdChainedScatterGatherService(

altServiceChainBuilder,

scatterGatherSupport,

partitionLoggingSupport

);

}

@Provides

@Singleton

Service<EarlybirdRequestContext, List<Future<EarlybirdResponse>>>

provideEarlybirdChainedScatterGatherService(

EarlybirdChainedScatterGatherService chainedScatterGatherService,

@Named(NAMED\_ALT\_CLIENT) @Nullable

EarlybirdChainedScatterGatherService altChainedScatterGatherService,

SearchDecider decider

) {

if (forceAltClientFlag.apply()) {

if (altChainedScatterGatherService == null) {

throw new RuntimeException(

"alt client cannot be null when 'force\_alt\_client' is set to true");

} else {

return altChainedScatterGatherService;

}

}

if (altChainedScatterGatherService == null) {

return chainedScatterGatherService;

}

return new SplitterService<>(

chainedScatterGatherService,

altChainedScatterGatherService,

decider,

ALT\_TRAFFIC\_PERCENTAGE\_DECIDER\_KEY

);

}

@Provides

@Singleton

@RecencyCache

Cache<EarlybirdRequest, EarlybirdResponse> provideRecencyCache(

JavaClient client,

DefaultForcedCacheMissDecider decider,

@Named(SearchRootModule.NAMED\_SERIALIZED\_KEY\_PREFIX) String serializedKeyPrefix,

@Named(SearchRootModule.NAMED\_CACHE\_KEY\_MAX\_BYTES) int cacheKeyMaxBytes,

@Named(SearchRootModule.NAMED\_CACHE\_VALUE\_MAX\_BYTES) int cacheValueMaxBytes) {

return EarlybirdCacheCommonModule.createCache(client, decider, CLUSTER + "\_recency\_root",

serializedKeyPrefix, TimeUnit.HOURS.toMillis(2), cacheKeyMaxBytes, cacheValueMaxBytes);

}

@Provides

@Singleton

@RelevanceCache

Cache<EarlybirdRequest, EarlybirdResponse> provideRelevanceCache(

JavaClient client,

DefaultForcedCacheMissDecider decider,

@Named(SearchRootModule.NAMED\_SERIALIZED\_KEY\_PREFIX) String serializedKeyPrefix,

@Named(SearchRootModule.NAMED\_CACHE\_KEY\_MAX\_BYTES) int cacheKeyMaxBytes,

@Named(SearchRootModule.NAMED\_CACHE\_VALUE\_MAX\_BYTES) int cacheValueMaxBytes) {

return EarlybirdCacheCommonModule.createCache(client, decider, CLUSTER + "\_relevance\_root",

serializedKeyPrefix, TimeUnit.HOURS.toMillis(2), cacheKeyMaxBytes, cacheValueMaxBytes);

}

@Provides

@Singleton

@StrictRecencyCache

Cache<EarlybirdRequest, EarlybirdResponse> provideStrictRecencyCache(

JavaClient client,

DefaultForcedCacheMissDecider decider,

@Named(SearchRootModule.NAMED\_SERIALIZED\_KEY\_PREFIX) String serializedKeyPrefix,

@Named(SearchRootModule.NAMED\_CACHE\_KEY\_MAX\_BYTES) int cacheKeyMaxBytes,

@Named(SearchRootModule.NAMED\_CACHE\_VALUE\_MAX\_BYTES) int cacheValueMaxBytes) {

return EarlybirdCacheCommonModule.createCache(client, decider, CLUSTER + "\_strict\_recency\_root",

serializedKeyPrefix, TimeUnit.HOURS.toMillis(2), cacheKeyMaxBytes, cacheValueMaxBytes);

}

@Provides

@Singleton

@TermStatsCache

Cache<EarlybirdRequest, EarlybirdResponse> provideTermStatsCache(

JavaClient client,

DefaultForcedCacheMissDecider decider,

@Named(SearchRootModule.NAMED\_SERIALIZED\_KEY\_PREFIX) String serializedKeyPrefix,

@Named(SearchRootModule.NAMED\_CACHE\_KEY\_MAX\_BYTES) int cacheKeyMaxBytes,

@Named(SearchRootModule.NAMED\_CACHE\_VALUE\_MAX\_BYTES) int cacheValueMaxBytes) {

return EarlybirdCacheCommonModule.createCache(client, decider, CLUSTER + "\_termstats\_root",

serializedKeyPrefix, TimeUnit.MINUTES.toMillis(5), cacheKeyMaxBytes, cacheValueMaxBytes);

}

@Provides

@Singleton

@TopTweetsCache

Cache<EarlybirdRequest, EarlybirdResponse> provideTopTweetsCache(

JavaClient client,

DefaultForcedCacheMissDecider decider,

@Named(SearchRootModule.NAMED\_SERIALIZED\_KEY\_PREFIX) String serializedKeyPrefix,

@Named(SearchRootModule.NAMED\_CACHE\_KEY\_MAX\_BYTES) int cacheKeyMaxBytes,

@Named(SearchRootModule.NAMED\_CACHE\_VALUE\_MAX\_BYTES) int cacheValueMaxBytes) {

return EarlybirdCacheCommonModule.createCache(client, decider, CLUSTER + "\_toptweets\_root",

serializedKeyPrefix, TopTweetsServicePostProcessor.CACHE\_AGE\_IN\_MS,

cacheKeyMaxBytes, cacheValueMaxBytes);

}

@Provides

SearchRootWarmup<EarlybirdService.ServiceIface, ?, ?> providesSearchRootWarmup(

Clock clock,

WarmupConfig config) {

return new EarlybirdWarmup(clock, config);

}

}